Attorney Docket No.: CXU-339

UNITED STATES PATENT APPLICATION

ENTITLED

DESIGNS FOR WIDE BAND ANTENNAS WITH PARASITIC ELEMENTS AND A METHOD TO OPTIMIZE THEIR DESIGN USING A GENETIC ALGORITHM AND FAST INTEGRAL EQUATION TECHNIQUE

BY

CHALMERS BUTLER
AND
SHAWN ROGERS

PATENT
Attorney Docket No: CXU-339

Designs for Wide Band Antennas with Parasitic Elements and a Method to Optimize Their Design Using a Genetic Algorithm and Fast Integral Equation Technique

PRIORITY CLAIM

This application claims the benefit of previously filed U.S. Provisional Application with the same titles and inventors as present, assigned USSN 60/215,434, filed on June 30, 2000, and which is incorporated herein by reference.

INTRODUCTION

This technology provides a method (application) of an algorithm to facilitate the design of wideband operations of antennas, and the design of sleeve cage monopole and sleeve helix, units. The technology is of interest/commercial potential throughout the audio communications community.

Omnidirectional capabilities and enhanced wideband capabilities are two desirable features for the design of many antenna applications. Designing omnidirectional antennas with wideband capabilities requires rapid resolution of complex relationship among antenna components to yield an optimal system. The invention comprises the use of a genetic algorithm with fitness values for design factors expressed in terms to yield optimum combinations of at least two types of antennas.

Cage antennas are optimized via a genetic algorithm (GA) for operation over a wide band with low voltage standing wave ratio (VSWR). Numerical results are compared to those of other dual band and broadband antennas from the literature. Measured results for one cage antenna are presented.

Genetic algorithms and an integral equation solver are employed to determine the position and lengths of parasitic wires around a cage antenna in order to minimize voltage standing wave ratio (VSWR) over a band. The cage is replaced by a normal mode quadrifilar helix for height reduction and the parasites are re-optimized. Measurements of the input characteristics of these optimized structures are presented along with data obtained from solving the electric field integral equation.

Genetic algorithms (Y. Rahmat-Samii and E. Michielssen, *Electromagnetic Optimizations by Genetic Algorithms*, New York: John Wiley and Sons, Inc., 1999) are used here in conjunction with an integral equation solution technique to determine the placement of the parasitic wires around a driven cage. The cage may be replaced by a quadrifilar helix operating in the normal mode in order to shorten the antenna. Measurements of these optimized structures are included for verification of the bandwidth improvements.

BACKGROUND AND SUMMARY OF THE INVENTION

Recent advances in modern mobile communication systems, especially those which employ spread-spectrum techniques such as frequency hopping, require antennas which have omnidirectional radiation characteristics, are of low profile, and can be operated over a very wide frequency range. The simple whip and the helical antenna operating in its normal mode appear to be attractive for this application because they naturally have omnidirectional characteristics and are mechanically simple. However, these structures are inherently narrow band and fall short of needs in this regard. Hence, additional investigations must be undertaken to develop methods to meet the wide bandwidth requirement of the communication systems.

This invention comprises a method to design (produce) a product and the product(s) designed/produced as a result of the application of the method. The products are broad band, omnidimensional communications antennas, and the design procedure involves the coordinated, sequential application of two algorithms: a generally described "genetic algorithm that simulates population response to selection and a new algorithm that is a fast wire integral equation solver that generates optimal multiple antenna designs from ranges of data that limit the end product. Individual designs comprise a population of designs upon which specified selection by limiting the genetic algorithm ultimate identifies the optimum design(s) for specified conditions. Superior designs so identified can be regrouped and a new population of designs generated for further selection/refinement.

The products are the antenna designs and specifications derived as a product of the application of the method briefly described above. The antennas all are characterized generally as broad band and omni directional, two features of critical importance in antenna design. In addition, although much of the theory has been developed on monopole antennas, both the method and designs include both monopole and dipole designs. In addition, the designs include sleeve-cage and sleeve-helix designs as hereinbelow further described.

The cage monopole comprises four vertical, straight wires connected in parallel and driven from a common stalk at the ground plane. The parallel straight wires are joined by crosses made of brass (or other conductive) strips, the width of which is equal to the electrical equivalent of the wire radius. Compared to a single wire, this cage structure has a lower peak voltage standing wave ratio (VSWR) over the band. A structure with lower VSWR is amenable to improved bandwidth characteristics with the addition of parasitic elements.

Adding parasitic elements of equal height and distance from the center of the cage monopole creates a sleeve cage monopole. The sleeve cage monopole has a greater bandwidth than its otherwise comparable antennas. Fitness values are determined by relative bandwidth, with greater fitness being associated with wider bandwidth defined by f2/f1, where f2 and f1 are respectively the largest and smallest frequencies between

which VSWR is 3.5 or less. Speed of optimization is increased by interpolation of the impedance matrix.

The heart of the process is the solution of the equation governing total axial current. The executable algorithm linked to the genetic algorithm by the fast wire integral equation solver provides a rapid method of solving this equation for varied values and inputs. The basic theory and equations are incorporated completely herein. See, S.D. Rogers and C.M. Butler, “An efficient curved-wire integral equation solution technique,” submitted to *IEEE Trans. Antennas Propagat.*

Reduced height without loss of bandwidth or omni-directional capabilities is a desired feature of antenna designs for a plurality of applications. These include installations in vehicles and confined interior spaces. The helix structure yields shorter antennas than the traditional whip structure with otherwise comparable features. Height is a function of the pitch angle of the helix, such that a pitch angle of 42 degrees reduces height by 30 percent. The addition of parasitic elements reduces VSWR in a helix configuration in a magnitude similar to the reduction noted for the cage monopole design.

Many modern wireless communication systems require low-profile antennas. To meet this requirement, we consider the helical antenna operating in the normal mode. A normal mode helix and a straight wire antenna having approximately the same wire length exhibit similar input impedance and far field patterns. One drawback to the helix operating in the normal mode is that its bandwidth is too limited for many applications. To increase the bandwidth, we have considered several potential remedies, one of which is discussed in this paper. It is well known that adding additional parasitic straight wires on either side of a driven dipole antenna may increase the bandwidth of the dipole (J.L. Wong and H.E. King, AP-21, no. 5, 725-727, Sept. 1973). One must be especially careful, however, to choose parasitic elements with the proper length and spacing. We use this basic idea to increase the bandwidth of the helical monopole. A structure similar to the sleeve dipole but applied to the helix has been used to design dual frequency antennas (P. Eratuuli, et. al., Electronics Letters, v. 32, no. 12, 1051-1052, June 1996). The helix and its helical sleeve are both driven in the antenna of this reference. Several novel antenna structures are considered such as a driven helical antenna adjacent to parasitic helices and straight wires. Another candidate structure consists of a driven helix with helical parasites inside or outside of the driven element, which has the added benefit of conserving space. In any case, due to the large number of parameters in a helix, it is more difficult to design a broadband sleeve helical antenna than is the case for a sleeve dipole. It is not feasible to obtain optimum values of parameters by trial and error. Thus we employ a genetic algorithm routine (D.L. Carroll, A FORTRAN Genetic Algorithm Driver, http://www.staff.uiuc.edu/~carroll/ga.html) and efficient integral equation solution techniques to optimize the antenna system for bandwidth. Having an efficient numerical solution technique is necessary for this problem since the geometry of the antenna is redefined for each structure evaluated by the genetic algorithm. Since these antennas have a high degree of curvature, their solutions generally require a large number of unknowns for representing the geometry. An efficient solution technique which gets

around this problem is used (S.D. Rogers and C.M. Butler, APS Symposium Digest, vol. I, 68-71, July 1997).

1) Commission B. B-2 Antennas

2) A genetic algorithm is used to optimize helical parasitic elements for a helical antenna.

3) This work is an extension to increasing the bandwidth of dipoles by use of parasites. This research could not have been completed in a time efficient manner without the development of an efficient integral equation solution technique for curved wires with reference below.

S.D. Rogers and C.M. Butler, "Reduced Rank Matrices for Curved Wire Structures," Digest of IEEE APS Symposium, Montreal, Canada, vol. I, pp. 68-71, July 1997.

We have recently shown from numerical calculations that the bandwidth of a normal mode helix can be increased by the addition of close-by wire parasites (S.D. Rogers, J.C. Young, and C.M. Butler, "Bandwidth Enhanced Normal Mode Helical Antennas," *Digest 1998 USNC / URSI National Radio Science Meeting*, Atlanta, GA., p. 293, June 1998). A genetic algorithm and a fast integral equation solution technique are employed to determine the optimum distance and height of these parasites. In (H.E. King and J.L. Wong, "An Experimental Study of a Balun-Fed Open-Sleeve Dipole in Front of a Metallic Reflector," *IEEE Trans. Antennas Propagat.* (Commun.), vol. AP-20, pp. 201-204, March 1972) these parameters were determined experimentally when the driven element was a straight wire dipole. In a recent paper (H. Nakano, et. al., "Realization of Dual-Frequency and Wide-Band VSWR Performances Using Normal-Mode Helical and Inverted-F Antennas," *IEEE Trans. Antennas Propagat.*, vol. AP-46, June 1998) a central parasitic straight cylinder was added inside a driven single wire helix to obtain dual frequency operation. We have found that even greater bandwidth, over that of a single driven wire, can be realized when the parasites are placed around "cage" monopoles having several parallel wires. Similar observations are made about a multifilament versus a single filament helix.

Bandwidth of vertically polarized wire antennas is often increased by adjustment of the antenna geometry. King and Wong reduce VSWR by placing parasitic wires around a driven element, creating the well-known open sleeve dipole (KING, H.E., and WONG, J.L.: 'An experimental study of a balun-fed open-sleeve dipole in front of a metallic reflector', *IEEE Trans. Antennas Propagat.*, March 1972, 20, (2), pp. 201-204). In (NAKANO, H., IKEDA, N., WU, Y., SUZUKI, R., MIMAKI, H., and YAMAUCHI, J.: 'Realization of dual-frequency and wide-band VSWR performances using normal-mode helical and inverted-F antennas', *IEEE Trans. Antennas Propagat.*, June 1998, 46, (6), pp. 788-793) the displacement of a parasitic monopole is varied inside a driven normal-mode helical antenna in order to control its characteristics. Cage antennas can be made broadband when their dimensions are chosen judiciously. Genetic algorithms and

integral equation solution techniques are employed here to optimize the dimensions of the cage antenna in order to create a structure with low VSWR over a wide band.

Recent advances in modern mobile communication systems, especially those which employ spread-spectrum techniques such as frequency hopping, require low-profile, broadband, omnidirectional (in azimuth) antennas. The simple whip and the helical antenna, operating in its normal mode, are potentially attractive for these applications because they naturally have suitable radiation characteristics and are mechanically simple and rugged. However, these structures are inherently narrow band. Hence, additional measures are employed to meet the wide bandwidth requirement of communication systems. Antennas often are loaded with tuning circuits and are connected to radios through matching networks in order to improve overall bandwidth. Altering the antenna geometry is another method for modifying bandwidth properties. The sleeve monopole, in which the outer conductor of the coaxial feed line forms a "sleeve" around the base of the protruding center conductor, is known to have greater bandwidth than the conventional monopole and has been studied extensively (J. Taylor, "The sleeve antenna," doctoral dissertation, Cruft Lab., Harvard Univ., Cambridge, MA, 1950); (R.W.P. King, The Theory of Linear Antennas. Cambridge, MA: Harvard Univ Press, 1956); (A.J. Poggio and P.E. Mayes, "Pattern bandwidth optimization of the sleeve monopole antenna," *IEEE Trans. Antennas Propagat.* (Commun.), vol. AP-14, pp. 643-645, Sept. 1966); (Z. Shen and R. MacPhie, "Rigorous evaluation of the input impedance of a sleeve monopole by modal-expansion method," *IEEE Trans. Antennas Propagat.*, vol. AP-44, pp. 1584-1591, Dec. 1996). A variation of this antenna is the open-sleeve dipole which has straight-wire parasites in place of the coaxial sleeve. The effects of the spacing and size of the parasitic elements on the VSWR are determined experimentally in (H.E. King and J.L. Wong, "An experimental study of a balun-fed open-sleeve dipole in front of a metallic reflector," *IEEE Trans. Antennas Propagat.* (Commun.), vol. AP-20, pp. 201-204, March 1972). In other papers, parasitic and driven elements of various sorts are combined in order to create dual band antennas. In (P. Eratuuli, *et. al.*, "Dual frequency wire antennas," *Electronics Letters*, vol. 32, no. 12, pp. 1051-1052, June 6, 1996) the driven wire is a straight monopole or a helix surrounded by a parasitic helix. In (H. Nakano, *et. al.*, "Realization of dual-frequency and wide-band VSWR performances using normal-mode helical and inverted-F antennas," *IEEE Trans. Antennas Propagat.*, vol. AP-46, pp. 788-793, June 1998) the position of a straight-wire parasite inside a driven normal mode helical antenna is adjusted to control the VSWR over the band of operation. Another antenna, which can be made to have broadband properties if its dimensions are chosen judiciously, is the cage antenna (S.D. Rogers and C.M. Butler, "Cage antennas optimized for bandwidth," submitted to *Electronics Letters*, April 2000). The cage is more amenable than a single straight wire to improvement in bandwidth when parasitic wires of appropriate size and spacing are added (S.D. Rogers and C.M. Butler, "The sleeve-cage monopole and sleeve helix for wideband operation," *Digest of APS Symposium*, Orlando Florida, vol. 2, pp. 1308-1311, July 1999).

We have found that the cage structure and multifilar helices are more amenable than single wire antennas to improvements in VSWR when parasitic wires are added. The

helical configuration can be used to reduce the height of the antenna, but at the sacrifice of bandwidth. While the addition of the parasitic wires improves the overall bandwidth, the VSWR increases outside the design band. Fast integral equation solution techniques and optimization methods have been developed in the course of this work and have led to effective tools for designing broadband antennas.

Certain exemplary attributes of the invention may relate to a method to create optimum design specifications for omni-directional, wide band antennas comprising the steps of:
(a) loading software including a genetic algorithm and an executable algorithm that is a fast wire equation solver into a computer;

(b) loading instructions into said computer specifying basic antenna design to be optimized;

(c) loading antenna design parameters and corresponding ranges of values for said parameters into said computer;

(d) specifying resolution of said parameters by loading number of bits per parameter into said computer;

(e) executing (operating) said genetic algorithm thereby generating a population of individual antenna designs each with a fitness value; and

(f) evaluating relative fitness of antenna designs produced and selecting superior designs for continued refinement.

The foregoing method may further comprise the following exemplary subroutines and algorithms for the software involved:

(a) a first algorithm that allows different values for critical design elements to combined in all possible combinations and a fitness value for each design ultimately estimated;

(b) a second algorithm that determines electronic current in an antenna by solving an integral equation numerically;

(c) a computer program link that provides essential communication between said first algorithm and said second algorithm.

Certain exemplary attributes of the invention may further relate to the sleeve monopole antenna designs, the cage sleeve monopole antenna designs, and the sleeve dipole antenna designs produced following the foregoing methods. Those of ordinary skill in the art will appreciate that various modifications and variations may be practiced in particular embodiments of the subject invention in keeping with the broader principles of the invention disclosed herein. The disclosures of all the citations herein referenced are fully incorporated by reference to this disclosure.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

DESIGN PROCEDURE

We modeled and measured the properties of a so-called cage monopole. The cage monopole shown in Figure 101a. consists of four vertical straight wires connected in parallel and driven from a common stalk at the ground plane. The ground plane in this model is assumed to be of infinite extent to facilitate analysis. The parallel straight wires are joined by crosses constructed of brass strips. The strip width was selected to be electrically equivalent to the wire radius for the purpose of modeling the structure. Compared to a single, thin, straight wire, the cage structure with multiple wires has a lower peak voltage standing wave ratio (VSWR) over the band. This is important since a structure which has a comparatively small VSWR over a band is more amenable to improvements in bandwidth with the addition of other components such as loads or parasites than is the common single-wire monopole with higher VSWR.

Next we add four parasitic straight wires of equal height (h) and distance (r) from the center of the cage to create the so-called "sleeve-cage monopole" of Figure 102a. The genetic algorithm of (D.L. Carroll, "A FORTRAN Genetic Algorithm Driver", Univ. of Illinois, Urbana, IL, http://www.staff.uiuc.edu/~carroll/ga.html) is used to determine the optimum distance and height of these parasitic straight wires. In this example the fitness value assigned to each antenna in the optimization process is the bandwidth ratio defined by f_2 / f_1, where f_2 and f_1 are, respectively, the largest and smallest frequencies between which the VSWR is 3.5 or less. We interpolate the impedance matrix with respect to frequency in order to increase the speed of the optimization process.

To design antennas that are smaller, we turn our attention to the normal mode helix, since, for operation about a given frequency, it can be made shorter than the vertical whip by adjustment of the pitch angle. Also, we observe a decrease in the peak VSWR when additional filaments are added to the helix driven from a central straight wire. Generally, a normal mode helix will exhibit electrical properties similar to those of a straight wire having the same wire length, though the peak VSWR for the helix is usually greater. The quadrifilar helix of Figure 103a whose height is 9.8 cm can be used in the same bands as a cage monopole of height about 14 cm. Thus, the total height of the antenna can be reduced by 30% with the 42° pitch angle. Parasitic straight wires of optimum height and distance are added to create what we call the "sleeve helical monopole" shown in Figure 104a.

RESULTS AND DISCUSSION

As one can see from the VSWR data, good agreement is achieved between predictions computed by means of our numerical techniques and results measured on a model mounted over a large ground plane. The frequency range over which data are presented is dictated by the frequencies over which our ground plane is electrically large. The slight discrepancies in the computed and measured results are attributed to imprecision in the construction of the antennas. The predicted results of bandwidth and VSWR of each antenna are summarized in the table below.

Structure	VSWR	BW Ratio	Frequency Range (MHZ)	Height (cm)	Width (cm)
Cage monopole	< 5.0	11.7:1	300-3500	17.2	2.2
	< 3.5	3:1	950-2850		
Sleeve-cage monopole	< 5.0	5.2:1	315-1650	17.2	5
	< 3.5	4.4:1	350-1550		
Quadrifilar helix	< 5.0	5.8:1	475-2750	9.8	2
	< 3.5	1.6:1	500-800		
Sleeve helix	< 5.0	3.9:1	475-1850	9.8	6
	< 3.5	3.5:1	500-1750		

We point out that when the parasitic elements are added to each structure, the bandwidth ratio increases for the VSWR < 3.5 requirement. However, outside of this frequency range the VSWR is worse than that of the antenna without parasites. In other words, VSWR has, indeed, been improved markedly over the design range but at a sacrifice in performance outside the range, where presumably the antenna would not be operated. Also, notice that the deep nulls in the directivity at the horizon for the cage and the quadrifilar helix structures have been eliminated with the addition of the parasites. Thus the directivity is improved in the band where on the basis of VSWR this antenna is deemed operable, although there was no constraint on directivity specified in the objective function.

CAGE ANTENNAS OPTIMIZED FOR BANDWIDTH

Design Method: The cage antenna is depicted in Fig. 201 where one sees four vertical wires joined to the feed and stabilized by thin brass strips of width w. The strips are treated as wires of radius $a = w/4$. The GA of (CARROLL, D.L.: 'Chemical Laser Modeling with Genetic Algorithms', *AIAA Journal*, Feb. 1996, 34, (2), pp. 338-346) is applied to optimize the diameter (d) of the cage structure and the length (h_2) of the wires in the cage. Each function evaluation consists of numerically solving the electric field integral equation for the cage geometry (having dimensions chosen by the GA) over the band of interest. Candidate antennas are given a fitness score equal to the bandwidth ratio f_h / f_l where f_l is the lowest and f_h is the highest frequency of operation over a band where the VSWR meets the design goal.

Results: The antenna of Fig. 202 is optimized for a design goal of VSWR < 2.0 over the frequency band 500 to 1600 MHz. The GA picks the parameter d from a range of 1 cm to 5 cm with a resolution of 0.13 cm (5 bits, 32 possibilities). The range specified for parameter h_2 is 8 cm to 12 cm with a resolution of 0.27 cm (4 bits, 16 possibilities). The GA converges to an optimum solution after three generations with five antennas per generation. A sensitivity analysis reveals that antenna input characteristics change only modestly with small geometric variation. The directivity of this cage antenna for $\phi = 0^\circ$ and $\theta = 75^\circ, 90^\circ$ is above 4 dBi over the entire band. The properties of this antenna and those of Nakano's helical monopole (NAKANO, H., IKEDA, N., WU, Y., SUZUKI, R.,

MIMAKI, H., and YAMAUCHI, J.: 'Realization of dual-frequency and wide-band VSWR performances using normal-mode helical and inverted-F antennas', *IEEE Trans. Antennas Propagat.*, June 1998, 46, (6), pp. 788-793), which is designed to operate with VSWR < 2.0 in two frequency bands, are listed in Table 201 for comparison.

The antenna of Fig. 203 is optimized for a design goal of VSWR < 2.5 in the frequency range 200 to 1200 MHz. This range is chosen for comparison of the cage antenna to the open sleeve dipole of (KING, H.E., and WONG, J.L.: 'An experimental study of a balun-fed open-sleeve dipole in front of a metallic reflector', *IEEE Trans. Antennas Propagat.*, March 1972, 20, (2), pp. 201-204) which operates over the frequency range 225 to 400 MHz. The GA is allowed to chose parameter d from 1 cm to 10 cm with a resolution of 0.6 cm (4 bits, 16 possibilities). The parameter h_2 is selected from 20 cm to 25 cm with a resolution of 0.33 cm (4 bits, 16 possibilities). An optimum result is reached after 11 generations with five antennas per generation. This cage monopole is not useful over the entire frequency range for which its VSWR is less than 2.5 since there is a null in the directivity within this range. It is operable over a 3.6:1 bandwidth for VSWR less than 2.5 and directivity greater than 0 dBi. In Table 201 are listed the properties of the cage antenna together with those of the sleeve dipole.

CAGE MONOPOLE AND SLEEVE-CAGE MONOPOLE

The cage monopole shown in Figure 301a consists of four vertical straight wires connected in parallel and driven from a common wire which is the extension of the center conductor of a coaxial cable protruding from the ground plane. The ground plane in this model is assumed to be of infinite extent in the analysis of the structure. The parallel straight wires are joined by crosses constructed of brass strips. The strip width w was selected to be electrically equivalent to the wire radius a for the purpose of modeling the structure ($w = 4a$) (C.M. Butler, "The equivalent radius of a narrow conducting strip," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp.755-758, July 1982). Compared to a single, thin, straight wire, the cage structure with multiple wires has a lower peak VSWR over the band as seen in Figure 301b. This is important since a structure which has a comparatively small VSWR over a band is more amenable to improvements in bandwidth with the addition of other components such as loads or parasites than is the common single-wire monopole with higher VSWR.

Four parasitic straight wires of equal height (h) and radial distance (r) from the center line of the cage are added to create the so-called "sleeve-cage monopole" of Figure 302a. The genetic algorithm of (D.L. Carroll, "Chemical Laser Modeling with Genetic Algorithms," *AIAA Journal*, vol. 34, no. 2), pp. 338-346, Feb. 1996) is used to determine optimum values of h and r for given design goals. An objective function evaluation for one antenna in the GA population involves numerically solving the electric field integral equation for many frequencies within the band of interest. Since this must be done for many candidate antennas, it is advantageous to interpolate the integral equation impedance matrix elements with respect to frequency (E.H. Newman, "Generation of wide-band data from the method of moments by interpolating the impedance matrix,"

IEEE Trans. Antennas Propagat., vol. AP-36, pp. 1820-1824, Dec. 1988). Each candidate structure is assigned a fitness value based on its electrical properties. A simple fitness value used here is the antenna bandwidth ratio which measures the performance of the antenna over a frequency band of interest denoted by $[f_A, f_B]$. The bandwidth ratio for a particular antenna is considered a function of its geometry and is computed from

$$F(h,r) = \frac{f_2}{f_1}$$

where

$$f_1 = \min_{f \in [f_A, f_B]}(f) \text{ such that } VSWR(f) \leq \text{limit}$$

and
$$f_2 = \max_{f \in [f_1, f_B]}(f) \text{ such that } VSWR(f) \leq \text{limit for all } f \in [f_1, f_2].$$

Another viable fitness value is the percent bandwidth defined here as

$$\%BW = 100\frac{f_2 - f_1}{\sqrt{f_2 f_1}}.$$

QUADRIFILAR HELIX AND SLEEVE HELIX

To design low profile antennas, we turn our attention to the normal mode helix, since, for operation about a given frequency, it can be made shorter than the vertical whip by adjustment of the helix pitch angle. Generally, a normal mode helix will exhibit electrical properties similar to those of a straight wire having the same wire length, though the peak VSWR for the helix is usually greater. The helix exhibits vertical polarization as long as it operates in the normal mode. There is a decrease in the peak VSWR, relative to that of a single-wire helix, when additional helical filaments are added to one driven from a central straight wire. The quadrifilar helix of Figure 303 whose height is 9.8 cm can be used in the same bands as a cage monopole of height about 14 cm. Thus, the total height of the antenna can be reduced by 30% with the 42° pitch angle. Parasitic straight wires of optimum height and distance are added to create what we call the "sleeve helical monopole" shown in Figure 304. Most integral equation solution techniques for the helix are, in general, more computationally expensive since these require many basis functions to represent the vector direction of the current along the meandering wire. A solution procedure which uncouples the representation of the geometry from the representation of the unknown current is developed in (S.D. Rogers and C.M. Butler, "An efficient curved-wire integral equation solution technique," submitted to *IEEE Trans. Antennas Propagat.)* and is used here to reduce the time in optimization of antennas with curved wires.

RESULTS

As one can see from the VSWR data, good agreement is achieved between predictions computed by means of numerical techniques (S.D. Rogers and C.M. Butler, "An efficient curved-wire integral equation solution technique," submitted to *IEEE Trans. Antennas Propagat.)* and results measured on a model mounted over a large ground plane. The frequency range over which our experiments are conducted is dictated by the frequencies over which the ground plane is electrically large. Of course, the dimensions of the antenna may be scaled for use in other bands. The slight discrepancies in the computed and measured results are attributed to the difficulty in building the antenna to precise dimensions. However, a sensitivity analysis reveals that the antenna performance changes minimally with small variations in geometry. The reflection coefficient is measured at the input of the coaxial cable driving the monopoles and of a shorted section of coaxial line having the same length. Applying basic transmission line theory to these data, one can determine the measured input impedance of the antenna with the reference "at the ground plane." All VSWR data is for a 50Ω system. As the feed point properties of the various antennas are evaluated, we must also keep in mind the radiation properties of the antenna, so computed directivity is included herein. The predicted results of bandwidth and VSWR of each antenna are summarized in Table 301.

Structure	VSWR	BW Ratio	BW %	Frequency Range (MHz)	Height (cm)	Width (cm)
Cage monopole	< 5.0	11.7	312	300-3500	17.2	2.2
	< 3.5	3	115	950-2850		
Sleeve-cage monopole	< 5.0	5.2	185	315-1650	17.2	5
	< 3.5	4.4	163	350-1550		
Quadrifilar helix	< 5.0	5.8	199	475-2750	9.8	2
	< 3.5	1.6	47	500-800		
Sleeve helix	< 5.0	3.9	147	475-1850	9.8	6
	< 3.5	3.5	134	500-1750		

Table 301 Summary of results.

We point out that, when the parasitic elements are added to each structure, the bandwidth ratio increases for the VSWR < 3.5 requirement. However, outside of this frequency range the VSWR is worse than that of the antenna without parasites. In other words, VSWR has, indeed, been improved markedly over the design range but at a sacrifice in performance outside the range, where presumably the antenna would not be operated. Also, notice that the deep nulls in the directivity at the horizon for the cage and the quadrifilar helix structures have been eliminated with the addition of the parasites. Thus the directivity is improved in the band where, on the basis of VSWR, this antenna is deemed operable, although there was no constraint on directivity specified in the objective function.

The following is a detailed description (including documentary references) of an exemplary efficient curved-wire integral equation solution technique as may be practiced in accordance with the subject invention.

AN EFFICIENT CURVED-WIRE INTEGRAL EQUATION SOLUTION TECHNIQUE

Shawn D. Rogers and Chalmers M. Butler
Department of Electrical and Computer Engineering, P.O. Box 340915, Clemson University
Clemson, SC 29634-0915

ABSTRACT

Computation of currents on curved wires by integral equation methods is often inefficient when the structure is tortuous but the length of wire is not large relative to wavelength at the frequency of operation. The number of terms needed in an accurate piecewise straight model of a highly curved wire can be large yet, if the total length of wire is small relative to wavelength, the current can be accurately represented by a simple linear function. In this paper a new solution method for the curved-wire integral equation is introduced. It is amenable to uncoupling of the number of segments required to accurately model the wire structure from the number of basis functions needed to represent the current. This feature lends itself to high efficiency. The principles set forth can be used to improve the efficiency of most solution techniques applied to the curved-wire integral equation. New composite basis and testing functions are defined and constructed as linear combinations of other commonly used basis and testing functions. We show how the composite basis and testing functions can lead to a reduced-rank matrix which can be computed via a transformation of a system matrix created from traditional basis and testing functions. Supporting data demonstrate the accuracy of the technique and its effectiveness in decreasing matrix rank and solution time for curved-wire structures.

I. INTRODUCTION

Numerical techniques for solving curved-wire integral equations [1] may involve large matrices, often due primarily to the resources needed to model the structure geometry rather than due to the number of basis functions needed to represent the unknown current. This is obviously true when a subdomain model is used to approximate a curvilinear structure in which the total wire length is small compared to the wavelength at the frequency of operation. Usually the number of segments needed in such a model is dictated by the structure curvature rather than by the number of weighted basis functions needed in the solution method to represent the unknown current. There is a demand for a general solution technique in which the number of unknowns needed to accurately represent the current is unrelated to the number of straight segments required to model (approximately) the meandering contour of the wire and the vector direction of the current. In recent years attention in the literature has been given to improving the numerical

efficiency of integral equation methods for curved-wire structures [2]-[13]. For the most part, presently available techniques incorporate basis functions defined on circular or curved wire segments. The authors of [2] define basis and testing functions along piecewise quadratic wire segments and achieve good results with fewer unknowns than would be needed in a piecewise straight model of a wire loop and of an Archimedian spiral antenna. Others introduce solution techniques for structures comprising circular segments that numerically model the current specifically on circular loop antennas [3], [4]. An analysis of general wire loops is presented in [5], where a Galerkin technique is employed over a parametric representation of a superquadric curve. In [6] arcs of constant radii are employed to define the geometry of arbitrarily shaped antennas from which is developed a technique for analyzing helical antennas. Other methods which utilize curved segments for subdomain basis and testing functions are available [7]-[10].

There are several advantages inherent in techniques in which basis and testing functions are defined over curved wire segments. Geometry modeling error can be made small and solution efficiency can be increased since to "fit" some structural geometries fewer curved segments are needed than is feasible with straight segments. Although these techniques are successful, they suffer disadvantages as well. First, the integral equation solution technique must be formulated to account for the new curved-segment basis and testing functions. This means that computer codes must be written to take advantage of the numerical efficiency of these new formulations incorporating the curved elements. A second disadvantage of curvilinear basis function modeling is that they fit one class of curve very well but are not well suited to structures comprising wires of mixed curvature. That is, circles fit loops and helices well but not spirals. Clearly, when a given structure comprises several arcs of different curvatures, the efficiency of methods employing a single curved-segment representation suffers. Elements like the quadratic segment or the arc-of-constant-radius segment increase the complexity of modeling. The third disadvantage of these techniques is that, for many structures, they do not lead to complete uncoupling of the number of the unknown current coefficients from the number of segments needed to model the structure geometry. For example, several quadratic segments or arcs, with one weighted unknown

defined on each, would be required to model the geometry of one turn of a multiturn helix, yet the current itself may be represented accurately in many cases by a simple linear function over several turns.

In this paper, an efficient method for solving for currents induced on curved-wire structures is presented. The solution method is based on modeling the curved wire by piecewise-straight segments but the underlying principles are general and can be exploited in conjunction with solution procedures which depend upon other geometry representations, including those that use arcs or curves. It is ideal for multi-curvature wire structures [12], [13]. The improved solution technique depends upon new basis and testing functions which are defined over more than two contiguous straight-wire segments. Composite basis functions are created as sums of weighted piecewise linear functions on wire segments, and composite testing functions compatible with the new basis functions are developed. The new technique allows one to reduce the rank of the traditional impedance matrix. We show how the matrix elements for a reduced-rank matrix can be computed from the matrix elements associated with a traditional integral equation solution method. Of paramount importance is the fact that the number of elements employed to model the geometric features of the structure is unrelated to the number of unknowns needed to accurately represent the wire current.

The concept of creating a new basis function as a linear combination of other basis functions is used in [14] for a multilevel iterative solution procedure for integral equations. Perhaps the composite basis function defined herein can be thought of as a "coarse level" basis function in multilevel terminology, although the method described in this paper is not related to the so-called multilevel or multigrid theory of [14]-[16].

The improved solution technique requires fewer unknowns than the traditional solution to represent the current on an Archimedian spiral antenna. Results comparable to those presented in [2] are achieved for the spiral. The improved technique also allows one to significantly reduce the number of unknowns required to solve for the current on wire helices. Specifically, the results of a convergence test show that the current on a helix can be modeled accurately with the same

number of unknowns needed for a "similar" straight wire even though the helix has a large number of turns.

II. INTEGRAL EQUATION FOR GENERAL CURVED WIRES

In this section we present the integro-differential equation governing the electric current on a general three dimensional curved or bent wire. Examples are the wire loop, the helix, and the meander line shown in Fig. 1. The wire is assumed to be a perfect electrical conductor and to be thin which means that the radius is much smaller than the wavelength and the length of wire. Under these thin-wire conditions the current is taken to be axially directed, circumferentially invariant, and zero at free ends. The equation governing the total axial current $I(s)\hat{\mathbf{s}}$ on the thin curved wire is

$$-j\frac{\eta}{4\pi k}\left\{k^2\int_C I(s')\hat{\mathbf{s}}'\cdot\hat{\mathbf{s}}K(s,s')ds' + \frac{d}{ds}\int_C \frac{d}{ds'}I(s')K(s,s')ds'\right\} = -\mathbf{E}^i(s)\cdot\hat{\mathbf{s}},\ s\in C \qquad (1)$$

in which C is the wire axis contour, s denotes the arc displacement along C from a reference to a point on the wire axis, and $\hat{\mathbf{s}}$ is the unit vector tangent to C at this point. The positive sense of this vector is in the direction of increasing s. $K(s,s')$ is the kernel or Green's function,

$$K(s,s') = \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{e^{-jkR}}{R}d\phi', \qquad (2)$$

in which R is the distance between the source and observation points on the wire surface, and $\mathbf{E}^i(s)$ is the incident electric field which illuminates the wire, evaluated in (1) on the wire surface at arc displacement s. Geometric parameters for an arbitrary curved wire are depicted in Fig. 2.

III. TRADITIONAL SOLUTION TECHNIQUE

The new solution method proposed in this paper can be viewed as an improvement to present methods. In fact, employing the ideas set forth in Section IV, one can modify an existing subdomain solution method to render it more efficient for solving the curved-wire integral

equation. Hence, the new method is explained in this paper as an enhancement of a method that has proved useful for a number of years. The method selected for this purpose is based on modeling the curved wire as an ensemble of straight-wire segments, with the unknown current represented as a linear combination of triangle basis functions and testing done with pulses. In this section this method is outlined as a basis for the explanation of the new method in Section IV.

The first step in modeling a curved wire is to select points on the wire axis and define vectors $\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_p$ from a reference origin to the selected points. The curved wire is modeled approximately as an ensemble of contiguous straight-wire segments joining these points (cf. Fig. 3). The arc displacement along the axis of the piecewise linear approximation of C is measured from the reference point labeled $\mathbf{r}_0$. The arc displacement between $\mathbf{r}_0$ and the n^{th} point located by $\mathbf{r}_n$ is l_n. A general point on the piecewise-straight approximation of the wire axis is located alternatively by means of the vector $\mathbf{r}$ and by the arc displacement l from the reference to the point. Various geometrical parameters describing the wire can be expressed in terms of the vectors locating the points on the wire axis. The unit vectors along the directions of the segments adjacent to the point $\mathbf{r}_p$ shown in Fig. 4 are given by

$$\hat{\mathbf{i}}_{p-} = \frac{\mathbf{r}_p - \mathbf{r}_{p-1}}{\Delta_{p-}} \tag{3}$$

$$\hat{\mathbf{i}}_{p+} = \frac{\mathbf{r}_{p+1} - \mathbf{r}_p}{\Delta_{p+}} \tag{4}$$

where

$$\Delta_{p-} = \left|\mathbf{r}_p - \mathbf{r}_{p-1}\right| \tag{5}$$

$$\Delta_{p+} = \left|\mathbf{r}_{p+1} - \mathbf{r}_p\right|. \tag{6}$$

The midpoint of the straight-wire segment joining $\mathbf{r}_p$ and $\mathbf{r}_{p\pm1}$ is located by

$$\mathbf{r}_{p\pm} = \frac{1}{2}\left[\mathbf{r}_p + \mathbf{r}_{p\pm 1}\right]. \tag{7}$$

In order to emphasize the fact that the model is now a straight wire segmentation of the original curved wire, s in (1) is replaced by l, the arc displacement along the axis of the straight wire model. With this notation and subject to the piecewise straight wire approximation, Eq. (1) becomes

$$-j\frac{\eta}{4\pi k}\left\{k^2\int_L I(l')\hat{\mathbf{l}}'\cdot\hat{\mathbf{l}}K(l,l')dl' + \frac{d}{dl}\int_L \frac{d}{dl'}I(l')K(l,l')dl'\right\} = -\mathbf{E}^i(l)\cdot\hat{\mathbf{l}},\ l\in L \tag{8}$$

where L is the piecewise straight approximation to C.

In a numerical solution of the integral equation for a curved wire structure, the (vector) current is expanded in a linear combination of weighted basis functions defined along the straight-wire segments. Even though they can be any of a number of functions, those employed here, for the purpose of illustration in this paper, are chosen to be triangle functions with support over two adjacent segments. Thus the current may be approximated by

$$I(l)\hat{\mathbf{l}}(l) \approx \sum_{n=1}^{N} I_n \Lambda_n(l)\hat{\mathbf{l}}_n(l) \tag{9}$$

in which the triangle function Λ_n about the n^{th} point on the segmented wire, as depicted in Fig. 5, is defined by

$$\Lambda_n(l) = \begin{cases} \dfrac{l-l_{n-1}}{\Delta_{n-}}, & l\in(l_{n-1},l_n) \\ \dfrac{l_{n+1}-l}{\Delta_{n+}}, & l\in(l_n,l_{n+1}) \end{cases} \tag{10}$$

where the unit vector $\hat{\mathbf{l}}_n$ is defined in terms of the unit vectors associated with the segments adjacent to the n^{th} point:

$$\hat{\mathbf{l}}_n = \begin{cases} \hat{\mathbf{l}}_{n-} \ , \ l \in (l_{n-1}, l_n) \\ \hat{\mathbf{l}}_{n+} \ , \ l \in (l_n, l_{n+1}) \end{cases}. \tag{11}$$

N is the number of basis functions and unknown current coefficients I_n in the finite series approximation (9) of the current. N unknowns are employed to represent the current on a wire having two free endpoints and modeled by $N+1$ straight-wire segments. In this traditional solution technique described here, N must be large enough to accurately model the geometric structure and vector direction of the current, *even if a large number of unknowns is not required to approximate the current* $I(l)$ *to the accuracy desired.* The triangle basis functions overlap as suggested in Fig. 6 so an approximation with N terms incorporates, at most, $N+1$ vector directions of current on the wire. These point-by-point directions of current on a curved wire must be accounted for accurately by the $N+1$ unit vectors, yet N piecewise linear basis functions may be far more than may be needed to accurately represent the current $I(l)$.

Testing the integro-differential equation is accomplished by taking the inner product of (8) with the testing function

$$\Pi_m(l) = \begin{cases} 1, & l \in (l_{m-}, l_{m+}) \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

depicted in Fig. 7 for $m = 1, 2, \ldots, N$. The inner product of this testing pulse with a function of the variable l is defined by

$$\langle f, \Pi_m \rangle = \int_{l_{m-}}^{l_{m+}} f(l)\,dl. \tag{13}$$

Expanding the unknown current I with (9) and taking the inner product of (8) with (12) for $m = 1, 2, \ldots, N$ yield a system of equations written in matrix form as

$$[Z_{mn}][I_n] = [V_m] \tag{14}$$

where

$$
\begin{aligned}
Z_{mn} = -j\frac{\eta}{4\pi k}\Bigg\{ & \frac{k^2}{2}\Big[\left(\Delta_{m-}\hat{\mathbf{i}}_{m-}\cdot\hat{\mathbf{i}}_{n-}+\Delta_{m+}\hat{\mathbf{i}}_{m+}\cdot\hat{\mathbf{i}}_{n-}\right)\int_{l_{n-1}}^{l_n}\Lambda_n(l')K(R_m)dl' \\
& +\left(\Delta_{m-}\hat{\mathbf{i}}_{m-}\cdot\hat{\mathbf{i}}_{n+}+\Delta_{m+}\hat{\mathbf{i}}_{m+}\cdot\hat{\mathbf{i}}_{n+}\right)\int_{l_n}^{l_{n+1}}\Lambda_n(l')K(R_m)dl'\Big] \\
& +\frac{1}{\Delta_{n-}}\int_{l_{n-1}}^{l_n}K(R_{m+})dl'-\frac{1}{\Delta_{n+}}\int_{l_n}^{l_{n+1}}K(R_{m+})dl'-\frac{1}{\Delta_{n-}}\int_{l_{n-1}}^{l_n}K(R_{m-})dl' \\
& +\frac{1}{\Delta_{n+}}\int_{l_n}^{l_{n+1}}K(R_{m-})dl'\Bigg\}
\end{aligned}
\tag{15}
$$

is an element of the $N\times N$ impedance matrix with

$$
R_m=\begin{cases}\sqrt{4a^2\sin^2\dfrac{\phi'}{2}+(l_m-l')^2}, & l_m \text{ and } l' \text{ on same segment}\\ \sqrt{|\mathbf{r}_m-\mathbf{r}'|^2+a^2}, & \text{otherwise}\end{cases}
\tag{16}
$$

and

$$
R_{m\pm}=\begin{cases}\sqrt{4a^2\sin^2\dfrac{\phi'}{2}+(l_{m\pm}-l')^2}, & l_{m\pm} \text{ and } l' \text{ on same segment}\\ \sqrt{|\mathbf{r}_{m\pm}-\mathbf{r}'|^2+a^2}, & \text{otherwise}\end{cases}.
\tag{17}
$$

When the source ($\mathbf{r}'$ or l') and observation ($\mathbf{r}$ or $l=(l_{m\pm},l_m)$) points reside on the same straight wire segment of radius a, as in Fig. 8 the exact kernel given by

$$
K(l,l')=\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{e^{-jkR}}{R}d\phi'
\tag{18}
$$

is used. Otherwise for source and observation points on different straight-wire segments (cf. Fig. 9), the exact kernel is approximated by the so-called reduced kernel,

$$
K(l,l')=\frac{e^{-jkR}}{R}.
\tag{19}
$$

The approximation below, which is excellent when the segment lengths are small compared with the wavelength, is employed in arriving at the first two terms of (15):

$$
\left\langle\hat{\mathbf{i}}(l)\cdot\mathbf{f}(l),\Pi_m(l)\right\rangle\approx\mathbf{f}(l_m)\cdot\left[\frac{1}{2}\Delta_{m-}\hat{\mathbf{i}}_{m-}+\frac{1}{2}\Delta_{m+}\hat{\mathbf{i}}_{m+}\right].
\tag{20}
$$

The same approximation can be used to compute the elements of the excitation column vector,

$$V_m = \left\langle -\mathbf{E}^i(l)\cdot\hat{\mathbf{l}}, \Pi_m \right\rangle \approx -\mathbf{E}^i(l_m)\cdot\left[\frac{1}{2}\Delta_{m-}\hat{\mathbf{l}}_{m-} + \frac{1}{2}\Delta_{m+}\hat{\mathbf{l}}_{m+}\right], \tag{21}$$

where $\mathbf{E}^i(l_m)$ is the known incident electric field at point l_m on the wire. Of course, if desired the left hand side of (21) can be evaluated numerically in those situations in which the incident field varies appreciably over a subdomain. We also point out that testing with pulses allows one to integrate directly the second term on the left side of (8). The derivative of the piecewise linear current in (8) leads to a pulse doublet (for charge) over two adjacent straight wire segments. These operations on the second term in the left side of (8) lead to the last four integrals in (15).

IV. IMPROVED SOLUTION TECHNIQUE

In this section a new technique for solving the curved-wire integral equation is presented. It is very efficient for tortuous wires on which the actual variation of the current is modest, a situation which often occurs when the length of wire in a given curve is small relative to wavelength, regardless of the degree of curvature. Composite basis and testing functions are introduced as an extension of the functions of the traditional solution method outlined in Section III. The composite basis function serves to uncouple the number of straight segments needed to model the curved-wire geometry and the vector direction of the current from the number of unknowns needed to accurately represent the current on the wire. This new basis function is a linear combination of appropriately weighted generic basis functions, *e.g.*, basis functions (9) in the traditional method outlined in Section III, and is defined over a number of contiguous straight segments. This new basis function is referred to as a composite basis function since it is constructed from others. Even though the solution method can incorporate any number of different generic basis and testing functions, the piecewise linear or triangle basis function and the pulse testing function are adopted here to facilitate explanation. Also, this pair leads to a very efficient and practicable solution scheme.

The notion of a composite triangle made up of constituent triangles is suggested in Fig. 10. For simplicity in illustration, the composite triangle is shown over a straight line though in

practice it would be over a polygonal line comprising straight-line segments, which approximate the curved wire axis. The q^{th} composite vector triangle function can be constructed as

$$\tilde{\Lambda}_q(l)\hat{\mathbf{l}}_q = \sum_{i=1}^{N^q} h_i^q \Lambda_i^q(l)\hat{\mathbf{l}}_i^q \tag{22}$$

in which Λ_i^q is the i^{th} constituent triangle defined by

$$\Lambda_i^q(l)\hat{\mathbf{l}}_i^q = \begin{cases} \dfrac{l-l_i^q}{\Delta_{i-}^q}\hat{\mathbf{l}}_{i-}^q, & l \in (l_{i-1}^q, l_i^q) \\ \dfrac{l_i^q - l}{\Delta_{i+}^q}\hat{\mathbf{l}}_{i+}^q, & l \in (l_i^q, l_{i+1}^q) \end{cases} \tag{23}$$

and illustrated in Fig. 11. When q is used as a superscript it identifies a parameter related to the q^{th} composite triangle function. The constitutive elements of the q^{th} composite basis function are denoted by the subscript i. The parameter h_i^q is the weight or magnitude of the i^{th} constituent triangle within $\tilde{\Lambda}_q$. These weights are functions of the segment lengths within each composite basis function and are adjusted so that the ordinate to the composite triangle is a linear function of displacement along the polygonal line which forms the base of the composite triangle. For example, for five constituent triangles in the q^{th} composite triangle of Fig. 10, the weights h_1^q and h_2^q are

$$h_1^q = \frac{\Delta_1^q}{\Delta_1^q + \Delta_2^q + \Delta_3^q} \tag{24}$$

$$h_2^q = \frac{\Delta_1^q + \Delta_2^q}{\Delta_1^q + \Delta_2^q + \Delta_3^q}. \tag{25}$$

The other weights are computed in a similar fashion. The parameter N^q is the number of triangle functions Λ_i^q employed to represent $\tilde{\Lambda}_q$. The example composite basis function of Fig. 10 is illustrated as the sum of five identical constituent triangles, but, of course, the constituents need not be the same if convenience or efficiency dictates otherwise. Also, this composite basis function is illustrated without the vector directions associated with each subdomain. In general the individual straight-wire segments over which a composite basis function is defined may each

have a different vector direction. Finally, the current expanded with a reduced number of unknowns $\widetilde{N}$ is

$$I(l)\hat{\mathbf{l}}(l)=\sum_{q=1}^{\widetilde{N}}\widetilde{I}_q\widetilde{\Lambda}_q(l)\hat{\mathbf{l}}_q(l) \tag{26}$$

where $\widetilde{\Lambda}_q(l)\hat{\mathbf{l}}_q(l)$ is the q^{th} vector composite basis function defined earlier in (22) and $\widetilde{I}_q$ is its unknown current coefficient. It is worth noting that constituent triangles are employed above to construct composite triangles but, if desired, they could be used to construct other basis functions, *e.g.*, an approximate, composite piecewise sinusoidal function.

If the number of unknowns in a solution procedure is reduced, then, of course, the number of equations must be reduced too which means that the testing procedure must be modified to achieve fewer equations. This is easily accomplished by defining composite testing pulses, compatible with the composite basis functions, as a linear combination of appropriately weighted constituent pulses. An example composite test pulse is depicted in Fig. 12. Such a p^{th} composite testing pulse is defined by

$$\widetilde{\Pi}_p(l)=\sum_{k=1}^{N^p}u_k^p\Pi_k^p(l) \tag{27}$$

where the constituent pulses associated with this p^{th} pulse are

$$\Pi_k(l)=\begin{cases}1, & l\in\left(l_{k-}^p,l_{k+}^p\right)\\ 0, & \text{otherwise}\end{cases} \tag{28}$$

and shown in Fig. 13. If with every constituent triangle there were associated a corresponding constituent pulse, then the testing functions $\widetilde{\Pi}_p$ would overlap, which is not desired and can be avoided by selecting the weight u_k^p to be 0 or 1 depending upon whether or not the k^{th} constituent pulse in $\widetilde{\Pi}_p$ is to be retained. To this end, the inner product of (13) is modified in the composite testing procedure to become

$$\left\langle f, \widetilde{\Pi}_p \right\rangle = \sum_{k=1}^{N^p} u_k^p \int_{l_{k-}^p}^{l_{k+}^p} f(l)\,dl\,. \tag{29}$$

Now that we have described the new basis and testing functions, we substitute the current expansion of (26) into (8) and form the inner product (29) of the resulting expression with $\widetilde{\Pi}_p$ for $p = 1,2,\ldots,\widetilde{N}$. This yields the following matrix equation having a reduced number ($\widetilde{N}$) of unknowns and equations:

$$\left[\widetilde{Z}_{pq}\right]\left[\widetilde{I}_q\right] = \left[\widetilde{V}_p\right] \tag{30}$$

where

$$\begin{aligned}
\widetilde{Z}_{pq} = \sum_{k=1}^{N^p} u_k^p \sum_{i=1}^{N^q} h_i^q \Bigg\{ -j\frac{\eta}{4\pi k}\Bigg[\frac{k^2}{2}\Bigg\{ &\left(\Delta_{k-}^p \hat{\mathbf{l}}_{k-}^p \cdot \hat{\mathbf{l}}_{i-}^q + \Delta_{k+}^p \hat{\mathbf{l}}_{k+}^p \cdot \hat{\mathbf{l}}_{i-}^q\right) \int_{l_{i-1}^q}^{l_i^q} \Lambda_i^q(l') K\left(R_k^p\right) dl' \\
&+ \left(\Delta_{k-}^p \hat{\mathbf{l}}_{k-}^p \cdot \hat{\mathbf{l}}_{i+}^q + \Delta_{k+}^p \hat{\mathbf{l}}_{k+}^p \cdot \hat{\mathbf{l}}_{i+}^q\right) \int_{l_i^q}^{l_{i+1}^q} \Lambda_i^q(l') K\left(R_k^p\right) dl' \Bigg\} \\
&+ \frac{1}{\Delta_{i-}^q} \int_{l_{i-1}^q}^{l_i^q} K\left(R_{k+}^p\right) dl' - \frac{1}{\Delta_{i+}^q} \int_{l_i^q}^{l_{i+1}^q} K\left(R_{k+}^p\right) dl' - \frac{1}{\Delta_{i-}^q} \int_{l_{i-1}^q}^{l_i^q} K\left(R_{k-}^p\right) dl' \\
&+ \frac{1}{\Delta_{i+}^q} \int_{l_i^q}^{l_{i+1}^q} K\left(R_{k-}^p\right) dl' \Bigg]\Bigg\}
\end{aligned} \tag{31}$$

represents an element of the reduced-rank $\left(\widetilde{N} \times \widetilde{N}\right)$ impedance matrix. At this point the reader is cautioned to distinguish between the index k which only appears in (31) as a subscript and the wave number $k = \omega\sqrt{\mu\varepsilon}$. The distances $R_{k\pm}^p$ and R_k^p are given in (16) or (17) with m replaced by index k, and the forcing function is given by

$$\widetilde{V}_p = -\sum_{k=1}^{N_p} u_k^p \int_{l_{k-}^p}^{l_{k+}^p} \mathbf{E}^i\left(l_k^p\right) \cdot \hat{\mathbf{l}}(l)\,dl\,.$$

One could compute the terms within the reduced-rank impedance matrix directly from (31). However, this would require more computation time than needed to fill the original impedance matrix of (14) since some constituent triangles within adjacent composite basis functions have the same support (Fig. 14). The constituent triangles within the overlapping portions of two adjacent composite basis functions differ only in the weight h_i^q. Therefore (31) incorporates redundancies which should be avoided. Also, a study of (15) and (31) reveals that the term within the braces of

(31) is identical to Z_{mn} of (15) if subscript i is replaced by n, subscript k by m, and the superscripts p and q are suppressed. Hence, the elements $\tilde{Z}_{pq}$ of the reduced-rank matrix can be computed from the elements Z_{mn} of the original matrix by means of the transformation

$$\tilde{Z}_{pq} = \sum_{k=1}^{N^p} u_k^p \sum_{i=1}^{N^q} h_i^q Z_{ki}^{pq} \tag{32}$$

where Z_{ki}^{pq} is a term in the original impedance matrix Z_{mn} of (15). The key to selecting appropriate Z_{mn} term is the combination of indices p, q, k, and i. The index p (q) indicates a group of rows (columns) in $[Z_{mn}]$ which are ultimately combined by the transformation in (32) to form the new matrix. The appropriate matrix element Z_{ki}^{pq} in $[Z_{mn}]$ is determined by intersecting the k^{th} row within the set of rows identified by index p with the i^{th} column of the group of columns specified by index q. Of course the groupings of rows and columns are determined when one defines the composite basis and testing functions.

A transformation for computing the reduced-rank matrix $[\tilde{Z}_{pq}]$ from the traditional matrix $[Z_{mn}]$ which is more efficient than is the construction of the matrix from (31) can be developed. The key to this transformation is (32). First, two auxiliary matrices $[L_{pm}]$ and $[R_{nq}]$ are constructed and, then, the desired transformation is expressed as

$$[\tilde{Z}_{pq}] = [L_{pm}][Z_{mn}][R_{nq}] \tag{33}$$

where

$$[L_{pm}] = \begin{bmatrix} u_1^1\, u_2^1 \cdots u_{N^1}^1 & & & \cdots & & & 0 \\ & u_1^2\, u_2^2 \cdots u_{N^2}^2 & & & & & \\ & & u_1^3\, u_2^3 \cdots u_{N^3}^3 & & & & \vdots \\ \vdots & & & \ddots & & & \\ & & & & u_1^p\, u_2^p \cdots u_{N^p}^p & & \\ & & & & & \ddots & \\ 0 & \cdots & & & & & u_1^{\tilde{N}}\, u_2^{\tilde{N}} \cdots u_{N^{\tilde{N}}}^{\tilde{N}} \end{bmatrix} \tag{34}$$

and

$$[R_{nq}] = \begin{bmatrix} h_1^1 & & & & & & \cdots \; 0 \\ h_2^1 & & & & & & \vdots \\ \vdots & h_1^2 & & & & & \\ h_{N^1}^1 & h_2^2 & & & & & \\ & \vdots & h_1^3 & & & & \\ & h_{N^2}^2 & h_2^3 & & & & \\ & & \vdots & & & & \\ & & h_{N^3}^3 & & & & \\ & & & \ddots & & & \\ & & & & h_1^q & & \\ & & & & h_2^q & & \\ & & & & \vdots & & \\ & & & & h_{N^q}^q & & \\ & & & & & \ddots & \\ & & & & & & h_1^{\tilde{N}} \\ & & & & & & h_2^{\tilde{N}} \\ \vdots & & & & & & \vdots \\ 0 & \cdots & & & & & h_{N^{\tilde{N}}}^{\tilde{N}} \end{bmatrix}. \tag{35}$$

It is easy to show that the above matrix transformation is equivalent to (32).

An alternative development of the transformation, which renders the meaning and construction of the matrices $[L_{pm}]$ and $[R_{nq}]$ more transparent is presented. We begin with the traditional N x N system matrix equation,

$$\left[Z_{mn}\right]\left[I_n\right]=\left[V_m\right], \tag{36}$$

which is to be transformed to the $\widetilde{N}$ x $\widetilde{N}$ reduced-rank matrix equation

$$\left[\widetilde{Z}_{pq}\right]\left[\widetilde{I}_q\right]=\left[\widetilde{V}_p\right]. \tag{37}$$

The number of unknown current coefficients in the original system of equations (36) is reduced by expressing the $\widetilde{N}$ coefficients $\widetilde{I}_q$ as linear combinations of the N coefficients I_n ($\widetilde{N}<N$). The $\widetilde{I}_q$ are constructed from the I_n by means of a scheme which accounts for the representation of the composite basis functions in terms of the original triangles on the structure. The resulting relationships among the original and the composite coefficients are expressed as

$$\left[I_n\right]=\left[R_{nq}\right]\left[\widetilde{I}_q\right] \tag{38}$$

where $\left[R_{nq}\right]$ embodies weights of the constituent triangles needed to synthesize composite basis function triangles. The matrix $\left[R_{nq}\right]$ directly combines unknown current coefficients consistent with the composite basis functions to result in a reduced number of unknowns. The construction is simple. If the triangle n from the original basis functions is to be used in the q^{th} composite basis function, the appropriate weight of this triangle is placed in row n and column q of $\left[R_{nq}\right]$. Otherwise zero is placed in this position. Again we point out that a given triangle may appear in more than one composite basis function. After substituting (38) into (36) we arrive at a modified system of linear equations

$$\left[Z_{mn}\right]\left[R_{nq}\right]\left[\widetilde{I}_q\right]=\left[V_m\right]. \tag{39}$$

which has a reduced number $\left(\widetilde{N}\right)$ of unknowns but the original number $\left(N\right)$ of equations. To reduce the number of equations to $\widetilde{N}$, tested linear equations are selectively added, which is accomplished by pre-multiplying (39) by $\left[L_{pm}\right]$ to arrive at

$$\left[L_{pm}\right]\left[Z_{mn}\right]\left[R_{nq}\right]\left[\widetilde{I}_q\right]=\left[L_{pm}\right]\left[V_m\right]. \tag{40}$$

The identifications,

$$\left[\widetilde{Z}_{pq}\right]=\left[L_{pm}\right]\left[Z_{mn}\right]\left[R_{nq}\right] \tag{41}$$

and

$$\left[\tilde{V}_p\right] = \left[L_{pm}\right]\left[V_m\right], \tag{42}$$

in (40) lead to the desired expression (37). The matrix $\left[L_{pm}\right]$ effectively creates composite testing functions from the original testing pulses. If the p^{th} composite testing pulse contains the m^{th} testing pulse from the original formulation, a one is placed in row p and column m of $\left[L_{pm}\right]$. Otherwise, a zero is placed in this position.

There are other important considerations in the implementation of this technique. Again, we label the number of basis functions in the traditional formulation N and the number of composite basis functions $\tilde{N}$. In the previous section the number of constituent triangles for the q^{th} composite basis function is designated N^q. Here for ease of implementation it is convenient to chose N^q to be the same value for every q, which we designate τ ($N^q = \tau$ for all q). Also, in the present discussion, we restrict τ to be one of the members of the arithmetic progression $5, 9, 13, 17, \ldots,$. With τ one of these integers, half-width constituent pulses are not required within the composite testing functions. N must be sufficiently large to ensure accurate modeling of the wire geometry and vector direction of the current as well as to preserve the numerical accuracy of the approximations. In addition, $\tilde{N}$ must be large enough to accurately represent the *variation* of the current. A convergence test must be conducted to arrive at acceptable values of N and $\tilde{N}$. Also, $N, \tilde{N}$ and τ must be defined carefully so that a value of τ in the arithmetic progression will allow an $N \times N$ matrix to be reduced to an $\tilde{N} \times \tilde{N}$ matrix. The following formula is useful for determining relationships between N and $\tilde{N}$, for a given value of τ, in the case of a general three-dimensional curved wire (without junctions):

$$\tilde{N} = 2\,\frac{N+1}{\tau+1} - 1\,. \tag{43}$$

For a wire structure with a junction, *e.g.,* a circular loop, where overlapping basis functions typically are used in the traditional formulation to satisfy Kirchhoff's current law, (43) becomes

$$\widetilde{N} = \frac{2N}{\tau + 1}. \tag{44}$$

Once N, $\widetilde{N}$ and τ are determined, it is easy to write a routine which determines the original basis and testing functions to be included in the composite functions. This information is then stored in the matrices $\left[L_{pm}\right]$ and $\left[R_{nq}\right]$.

In the above, composite triangle expansion functions are synthesized from generic triangle functions but one could as well, if desired, approximate other composite expansion functions, *e.g.*, "sine triangles" by adjustment of the coefficients h_i^q. Similarly, other approximate testing functions could be created by adjustment of the factors u_k^p. Thus, a reduced-rank solution method with composite expansion and testing functions different from triangles and pulses could be readily created from the techniques discussed in this section. Only h_i^q and u_k^p, peculiar to the functions selected in the method to be implemented, must be changed in (32) in order to arrive at the appropriate reduced-rank matrix elements $\widetilde{Z}_{pq}$. If $\left[L_{pm}\right]$ of (34) were replaced by $\left[R_{nq}\right]^T$ in (33) where $\left[R_{nq}\right]$ is defined in (35) and T denotes transpose, then the resulting reduced-rank matrix $\left[\widetilde{Z}_{pq}\right]$ would be that for a method which employs composite triangle expansion and (approximate) composite triangle testing functions.

V. RESULTS

Results obtained by solving the integral equation of (15) with the improved solution method developed above are presented in this section as are values of current determined by the traditional method. In some cases data obtained from the literature are displayed for comparison. Results are presented for the wire loop, an Archimedian spiral antenna, and several different helical antennas and scatterers.

Current values on a small wire loop antenna are depicted in Fig. 15. The loop is modeled by 32 linear segments (and 32 unknowns) in the traditional solution technique. Also shown are values obtained from the new solution method with eight composite basis functions (eight unknowns) each having five constituent triangles constructed on twenty four linear segments.

These current values compare well with those from the traditional solution and with data from [2] where the loop is modeled with eight unknowns on quadratic segments. There is slight disagreement at the driving point which is to be expected (with eight unknowns) near a delta gap source where the current varies markedly. To investigate this discrepancy we use three triangle basis functions in the vicinity of the delta-gap source and do not form composite triangles in this region. The results are shown in Fig. 16. Here the loop problem has been solved with 28 unknowns for the traditional method and twelve unknowns for the composite basis function solution. It is seen that the agreement is excellent even in the vicinity of the delta-gap source.

The improved solution method is applied to a four arm Archimedian spiral antenna. This antenna is chosen since it is used in [2] to illustrate the usefulness of the quadratic subdomains for wires having significant curvature. A description of the geometry of Archimedian spiral antennas is found in [17] and [18]. The antenna is excited by a delta gap source on each arm located near the junction of the four arms. The results presented in this section are for mode 2 excitation [19]. The antenna is also modeled by the traditional technique with 725 unknowns on each arm (725*4+3=2903). In [17] the authors implement a discrete body of revolution technique so that the number of unknowns needed for one arm is sufficient for solving the problem. Since our goal is to employ the data of [17] to demonstrate the accuracy of our method and not to create the best analytical tool for the Archimedian spiral antenna, we solve this problem by including the same number of linear segments on each arm and placing overlapping triangles at the wire junction to enforce Kirchhoff's current law. In [2] it is found that each arm requires 504 linear segments to obtain an accurate solution. They also obtain accurate values of the current with 242 quadratic segments. We reproduce these results with our improved solution method as illustrated in Fig. 17-Fig. 19. The number of unknowns for each arm is 725 for the traditional technique and 241 for the improved method. In each composite basis function there are five constituent triangles. In Fig. 17 the difference in the solution of the current for the two methods is seen to be negligible. Good agreement is also achieved for the current magnitude (cf. Fig. 18). A favorable comparison with data from [2] is observed in Fig. 19. Since the symmetry in the geometry is not

used to further reduce the number of unknowns required for the structure, the actual number of unknowns in the impedance matrices are 2903 and 967, respectively. The computation times for the various routines of the FORTRAN 90 code are presented in the table below. All times are for runs on a 375 MHz DEC Alpha processor. The time study shows that the reduction technique is successful in significantly reducing matrix solve time for this four-arm Archimedian spiral antenna. A standard linear equation solution method is employed to solve both sets of linear equations since the objective of this comparison is to delineate the enhanced efficiency of the reduced-rank method.

TABLE I

COMPUTATION TIMES FOR ARCHIMEDIAN SPIRAL

Event	Time in Seconds
Fill matrix N=2903	1020
Solve matrix equation N=2903	1329
Reduce matrix from 2903 to 967	5.54
Solve reduced matrix equation N=967	45.81
Traditional method total time	2349
Improved method total time	1071

Consider next a ten-turn helix having a total wire length of 0.5λ and illuminated by a plane wave. The geometry of the helical scatterer is depicted in Fig. 20. The current shown in Fig. 21 is "converged" when the number of unknowns in the traditional solution technique reaches 259. Thus one concludes that 260 linear segments are required to accurately represent the geometry of this structure and vector nature of the current. We determine convergence by examining the real and imaginary parts of the current along the structure. When changes in the current are sufficiently small as the number of segments is increased, convergence is assumed [2]. The results of a convergence test show that an accurate solution of the current can be achieved with 51 composite basis functions. The number of constituent triangles in each basis function in this case is nine. We note that the solution with 27 composite basis functions differs only slightly from the converged solution.

The current is shown in Fig. 22 for another helical scatterer of geometry similar to that described above and subject to the same excitation and geometry similar to that described above. The circumference of each turn of this ten-turn helix is 0.035λ making the total wire length 0.35λ. These results are given as an example to illustrate that the composite basis function scheme works well with curved-wire structures having a wire length which is not an integer multiple of half wavelength.

The data of Fig. 23 are for a 50-turn helix having a total wire length of 2λ and illuminated by a plane wave traveling in the positive x direction. One sees that 27 unknowns are adequate to accurately represent the current along the helix. However, 1483 unknowns are required in the traditional solution method since many linear segments are required to define the 50-turn structure and the vector properties of the current. In this example there are 105 constituent triangles in each composite basis function. The table below shows the computational savings enjoyed by the method of this paper.

TABLE II

COMPUTATION TIMES FOR FIFTY-TURN

HELIX

Event	Time in Seconds
Fill matrix N=1483	300
Solve matrix equation N=1483	267
Reduce matrix rank from 1483 to 27	1.84
Solve reduced matrix equation N=27	Negligible
Traditional method total time	567
Improved method total time	302

Next we illustrate the prowess of the solution technique for helical antennas. Specifically the data presented in Fig. 24 and Fig. 25 are for helical antennas driven above a ground plane by a delta gap source. The geometry of the helix is given in Fig. 20 and the ground plane is located at $z = 0$. The data of the improved method compare well with those of the traditional solution technique, but, again, there is a slight difference in the currents at the ground plane due to the nature of the delta gap source. In each of these figures the number of unknowns given is the

number for the structure plus its image, but data are plotted only for the part of the structure above the ground plane. Since there are many turns, the number of segments needed to represent the geometry of the antenna and its image is large. The number of unknowns is reduced from N=917 in the traditional method to N=53 in the improved technique. Of course, one could employ image theory to modify the integral equation which could be solved by the new method with an even more dramatic savings in computer resources.

The last example is a five-turn helical antenna over an infinite ground plane, driven by a delta gap source. This structure is included here because it is used in [6] to exhibit the accuracy of a technique employing basis and testing functions defined over arcs of constant radii. It is modeled by straight wire segments in [20]. In [6] the authors discretize the antenna into fifteen arcs and then compare solutions of 135 unknowns with forty-five unknowns. They find that forty-five unknowns is enough to obtain an accurate solution for the current when the geometry is defined by arcs. We reproduce these results except that the antenna geometry is defined by many straight wire segments. In the method of this paper we include the unknowns on the image (269 unknowns on the antenna plus its image corresponds to 135 unknowns on the antenna above the ground plane). Likewise, 89 unknowns on the antenna and image are equivalent to 45 unknowns on the antenna. We find that helical antennas require a minimum of 25 unknowns per turn in the traditional solution technique in order to represent the geometry. In order to reduce the number of unknowns over the antenna and its image from 269 to 89, each composite basis function is constructed with 5 constituent triangles. A qualitative comparison of our data and that of [6] suggests agreement in the two methods.

VI. CONCLUSIONS

The solution method presented in this paper is very simple and practicable for reducing the rank of the impedance matrix for curved-wire structures. It should be mentioned that rank reduction is realized only when the number of segments needed to model the geometry and vector direction of the current exceeds the number of unknown current coefficients necessary to

characterize the variation of the current. We define composite basis and testing functions as the sum of constituents over linear segments on a wire and arrive at a new impedance matrix of reduced rank. It is shown how this reduced-rank matrix can be determined from the original impedance matrix by a matrix transformation. Thus one advantage of this technique is that it can be applied to almost any existing curved-wire codes which define basis and testing functions over straight-wire segments or curved-wire segments.

Dramatic savings in matrix solve time are realized for the cases of the four-arm Archimedian spiral antenna and the helical antenna. The benefits for reducing unknowns on, for example, a helical antenna become much more significant as the number of turns increases. It should be pointed out that this method does not reduce matrix fill time since the elements of the original impedance matrix are computed as a step in the determination the elements of the reduced-rank matrix. Problems involving large curved-wire structures can be solved readily by this method, *e.g.,* a straight wire antenna loaded with multiple, tightly wound helical coils and an array of Archimedian spiral antennas. The principles described here can be used in addition to other methods such as those based upon iteration.

REFERENCES

[1] R.F. Harrington, *Field Computation by Moment Methods*. Malabar, FL: Krieger, 1968.

[2] N.J. Champagne, II, J.T. Williams, and D.R. Wilton, "The use of curved segments for numerically modeling thin wire antennas and scatterers," *IEEE Trans. Antennas Propagat.*, vol. 40, pp. 682-689, June 1992.

[3] E.K.N. Yung and R.S.K.Wong, "Analysis of an array of circular loops." *Annals of Telecommunications*, vol. 48, no. 9-10, pp. 491-497, 1993.

[4] E.K.N. Yung and R.S.K.Wong, "Analysis of a thin wire circular loop antenna," *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 7, pp. 189-199, 1994.

[5] M.A. Jensen and Y. Rahmat-Samii, "Electromagnetic characteristics of superquadric wire loop antennas," *IEEE Trans. Antennas Propagat.*, vol. 42, pp. 264-269, February 1997.

[6] E.K.N. Yung and R.S.K.Wong, "Analysis of a wire antenna of arbitrary shape," *J. Electromagnetic Waves and Applications*, vol. 9, no. 7/8, pp. 855-869, 1995.

[7] S.K. Khamas, G.G. Cook, and R.J. Waldron, "Moment-Method analysis of printed circular wire-loop antenna using curved piecewise sinusoidal subdomain basis and test functions," *IEEE Trans. Antennas Propagat.*, vol. 44, pp. 1303-1305, Sept. 1996.

[8] S.K. Khamas, G.G. Cook, "Moment-Method analysis of printed wire spirals using curved piecewise sinusoidal subdomain basis and test functions," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 1016-1022, June 1997.

[9] G.G. Cook and S.K. Khamas, "Efficient moment method for analysing printed wire loop antennas," *IEE Proc.-Microw. Antennas Propag.*, vol. 144, no.5, pp.364-366, October 1997.

[10] S.K. Khamas, et. al., "Moment method analysis of printed single-arm wire spiral antennas using curved segments," *IEE Proc.-Microw. Antennas Propag.*, vol. 144, no. 4., pp. 261-265, August 1997.

[11] B.M. Kolundzija and B.D. Popovic, "Entire domain Galerkin method for analysis of generalised wire antennas and scatterers," *IEE Proc.-Microw. Antennas Propag.*, vol. 139, no. 1 pp. 17-24, Feb. 1992.

[12] S.D. Rogers and C.M. Butler, "Reduced Rank Matrices for Curved Wire Structures," Digest of IEEE APS/URSI Radio Science Meeting, July 1997.

[13] S.D. Rogers, "Efficient Numerical Techniques for Curved Wires," M.S.E.E. thesis, Clemson University, Clemson, SC, 1997.

[14] K. Kalbasi and K.R. Demarest, "A Multilevel Formulation of the Method of Moments," *IEEE Trans. Antennas Propagat.*, vol. 41, pp. 589-599, May 1993.

[15] J.H. Bramble, *Multigrid Methods*. New York: John Wiley and Sons, 1993.

[16] S.F. McCormick, *Multigrid Methods: Theory, Applications, and Supercomputing*. New York: Marcel Dekker, 1988.

[17] N.J. Champagne, II, "Method of moments formulations for thin wire antennas and scatterers using piecewise linear and curved wire segments," M.S.E.E. thesis, Univ. Houston, Houston, TX, 1991.

[18] N.J. Champagne, II, J.T. Williams, R.M. Sharpe, S.U. Hwu, and D.R. Wilton, "Numerical modeling of impedance loaded multi-arm Archimedian spiral antennas," *IEEE Trans. Antennas Propagat.*, vol. 40, pp. 102-108, Jan. 1992.

[19] H. Nakano, J. Yamauchi, and S. Hashimoto, "Numerical Analysis of 4-Arm Archimedian spiral antenna," *Electron. Lett.*, vol. 19, no. 3, pp. 78-80, 1983.

[20] H. Nakano, *Helical and Spiral Antennas*, John Wiley & Sons, Inc, New York, pp. 122-126, 1987.

[21] H. Nakano, S.R. Kerner, A.G. Alexopoulos, "The moment method solution for printed wire antennas of arbitrary configuration," *IEEE Trans. Antennas Propagat.*, vol. 36, pp.1667-1674, Dec. 1988.

[22] M. Ali, S.S. Stuchly, and K. Caputa, "Characteristics of bent wire antennas," *J. Electromagnetic Waves and Applications*, vol. 9, no. 9, pp. 1149-1162, 1995.

[23] K.K. Mei, "On the Integral Equations of Thin Wire Antennas," *IEEE Trans. Antennas Propagat.*, vol. 13, pp. 374-378, May 1965.

The following is a detailed description of an exemplary genetic algorithm that can be used in accordance with the subject invention to obtain optimal antenna parameters for given design criteria.

```
c#######################################################################
c
      program gafortran
c
c  This is version 1.7, last updated on 12/11/98.
c
c  Copyright David L. Carroll; this code may not be reproduced for sale
c  or for use in part of another code for sale without the express
c  written permission of David L. Carroll.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  David L. Carroll
c  University of Illinois
c  306 Talbot Lab
c  104 S. Wright St.
c  Urbana, IL  61801
c
c  e-mail: carroll@uiuc.edu
c  Phone:  217-333-4741
c  fax:    217-244-0720
c
c  This genetic algorithm (GA) driver is free for public use.  My only
c  request is that the user reference and/or acknowledge the use of this
c  driver in any papers/reports/articles which have results obtained
c  from the use of this driver.  I would also appreciate a copy of such
c  papers/articles/reports, or at least an e-mail message with the
c  reference so I can get a copy.  Thanks.
c
c  This program is a FORTRAN version of a genetic algorithm driver.
c  This code initializes a random sample of individuals with different
c  parameters to be optimized using the genetic algorithm approach, i.e.
c  evolution via survival of the fittest.  The selection scheme used is
c  tournament selection with a shuffling technique for choosing random
c  pairs for mating.  The routine includes binary coding for the
c  individuals, jump mutation, creep mutation, and the option for
c  single-point or uniform crossover.  Niching (sharing) and an option
c  for the number of children per pair of parents has been added.
c  An option to use a micro-GA is also included.
c
c  For companies wishing to link this GA driver with an existing code,
c  I am available for some consulting work.  Regardless, I suggest
c  altering this code as little as possible to make future updates
c  easier to incorporate.
c
c  Any users new to the GA world are encouraged to read David Goldberg's
c  "Genetic Algorithms in Search, Optimization and Machine Learning,"
c  Addison-Wesley, 1989.
c
c  Other associated files are:  ga.inp
c                               ga.out
c                               ga.restart
c                               params.f
c                               ReadMe
c                               ga2.inp (w/ different namelist identifier)
c
c  I have provided a sample subroutine "func", but ultimately
c  the user must supply this subroutine "func" which should be your
c  cost function.  You should be able to run the code with the
c  sample subroutine "func" and the provided ga.inp file and obtain
c  the optimal function value of 1.0000 at generation 187 with the
c  uniform crossover micro-GA enabled (this is 935 function evaluations).
c
```

```
c  The code is presently set for a maximum population size of 200,
c  30 chromosomes (binary bits) and 8 parameters.  These values can be
c  changed in params.f as appropriate for your problem.  Correspondingly
c  you will have to change a few 'write' and 'format' statements if you
c  change nchrome and/or nparam.  In particular, if you change nchrome
c  and/or nparam, then you should change the 'format' statement numbers
c  1050, 1075, 1275, and 1500 (see ReadMe file).
c
c  Please feel free to contact me with questions, comments, or errors
c  (hopefully none of latter).
c
c  Disclaimer:  this program is not guaranteed to be free of error
c  (although it is believed to be free of error), therefore it should
c  not be relied on for solving problems where an error could result in
c  injury or loss.  If this code is used for such solutions, it is
c  entirely at the user's risk and the author disclaims all liability.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension fitness(indmax),nposibl(nparmax),nichflg(nparmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension ibest(nchrmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
      dimension geni(1000000),genavg(1000000),genmax(1000000)
c      real*4 cpu,cpu0,cpu1,tarray(2)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common / ga8   / nichflg
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Input variable definitions:
c
c  icreep   = 0 for no creep mutations
c           = 1 for creep mutations; creep mutations are recommended.
c  idum     The initial random number seed for the GA run.  Must equal
c           a negative integer, e.g. idum=-1000.
c  ielite   = 0 for no elitism (best individual not necessarily
c               replicated from one generation to the next).
c           = 1 for elitism to be invoked (best individual replicated
c               into next generation); elitism is recommended.
c  iend         = 0 for normal GA run (this is standard).
c           = number of last population member to be looked at in a set
c             of individuals.  Setting iend-0 is only used for debugging
c             purposes and is commonly used in conjunction with iskip.
c  iniche   = 0 for no niching
c           = 1 for niching; niching is recommended.
c  irestrt  = 0 for a new GA run, or for a single function evaluation
c           = 1 for a restart continuation of a GA run.
```

```
c  iskip    = 0 for normal GA run (this is standard).
c           = number in population to look at a specific individual or
c             set of individuals.  Setting iskip-0 is only used for
c             debugging purposes.
c  itourny  No longer used.  The GA is presently set up for only
c           tournament selection.
c  iunifrm  = 0 for single-point crossover
c           = 1 for uniform crossover; uniform crossover is recommended.
c  kountmx  = the maximum value of kount before a new restart file is
c             written; presently set to write every fifth generation.
c             Increasing this value will reduce I/O time requirements
c             and reduce wear and tear on your storage device
c  maxgen   The maximum number of generations to run by the GA.
c           For a single function evaluation, set equal to 1.
c  microga  = 0 for normal conventional GA operation
c           = 1 for micro-GA operation (this will automatically reset
c             some of the other input flags).  I recommend using
c             npopsiz=5 when microga=1.
c  nchild   = 1 for one child per pair of parents (this is what I
c               typically use).
c           = 2 for two children per pair of parents (2 is more common
c               in GA work).
c  nichflg  = array of 1/0 flags for whether or not niching occurs on
c             a particular parameter.  Set to 0 for no niching on
c             a parameter, set to 1 for niching to operate on parameter.
c             The default value is 1, but the implementation of niching
c             is still controlled by the flag iniche.
c  nowrite  = 0 to write detailed mutation and parameter adjustments
c           = 1 to not write detailed mutation and parameter adjustments
c  nparam   Number of parameters (groups of bits) of each individual.
c           Make sure that nparam matches the number of values in the
c           parmin, parmax and nposibl input arrays.
c  npopsiz  The population size of a GA run (typically 100 works well).
c           For a single calculation, set equal to 1.
c  nposibl  = array of integer number of possibilities per parameter.
c             For optimal code efficiency set nposibl=2**n, i.e. 2, 4,
c             8, 16, 32, 64, etc.
c  parmax   = array of the maximum allowed values of the parameters
c  parmin   = array of the minimum allowed values of the parameters
c  pcreep   The creep mutation probability.  Typically set this
c           = (nchrome/nparam)/npopsiz.
c  pcross   The crossover probability.  For single-point crossover, a
c           value of 0.6 or 0.7 is recommended.  For uniform crossover,
c           a value of 0.5 is suggested.
c  pmutate  The jump mutation probability.  Typically set = 1/npopsiz.
c
c
c  For single function evaluations, set npopsiz=1, maxgen=1, & irestrt=0.
c
c  My favorite initial choices of GA parameters are:
c     microga=1, npopsiz=5, iunifrm=1, maxgen=200
c     microga=1, npopsiz=5, iunifrm=0, maxgen=200
c  I generally get good performance with both the uniform and single-
c  point crossover micro-GA.
c
c  For those wishing to use the more conventional GA techniques,
c  my old favorite choice of GA parameters was:
c     iunifrm=1, iniche=1, ielite=1, itourny=1, nchild=1
c  For most problems I have dealt with, I get good performance using
c     npopsiz=100, pcross=0.5, pmutate=0.01, pcreep=0.02, maxgen=26
c  or
c     npopsiz= 50, pcross=0.5, pmutate=0.02, pcreep=0.04, maxgen=51
c
c  Any negative integer for idum should work.  I typically arbitrarily
```

```
c  choose idum=-10000 or -20000.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Code variable definitions (those not defined above):
c
c  best      = the best fitness of the generation
c  child     = the floating point parameter array of the children
c  cpu       = cpu time of the calculation
c  cpu0,cpu1= cpu times associated with 'etime' timing function
c  creep     = +1 or -1, indicates which direction parameter creeps
c  delta     = del/nparam
c  diffrac   = fraction of total number of bits which are different
c              between the best and the rest of the micro-GA population.
c              Population convergence arbitrarily set as diffrac<0.05.
c  evals     = number of function evaluations
c  fbar      = average fitness of population
c  fitness   = array of fitnesses of the parents
c  fitsum    = sum of the fitnesses of the parents
c  genavg    = array of average fitness values for each generation
c  geni      = generation array
c  genmax    = array of maximum fitness values for each generation
c  g0        = lower bound values of the parameter array to be optimized.
c              The number of parameters in the array should match the
c              dimension set in the above parameter statement.
c  g1        = the increment by which the parameter array is increased
c              from the lower bound values in the g0 array.  The minimum
c              parameter value is g0 and the maximum parameter value
c              equals g0+g1*(2**g2-1), i.e. g1 is the incremental value
c              between min and max.
c  ig2       = array of the number of bits per parameter, i.e. the number
c              of possible values per parameter.  For example, ig2=2 is
c              equivalent to 4 (=2**2) possibilities, ig2=4 is equivalent
c              to 16 (=2**4) possibilities.
c  ig2sum    = sum of the number of possibilities of ig2 array
c  ibest     = binary array of chromosomes of the best individual
c  ichild    = binary array of chromosomes of the children
c  icount    = counter of number of different bits between best
c              individual and other members of micro-GA population
c  icross    = the crossover point in single-point crossover
c  indmax    = maximum # of individuals allowed, i.e. max population size
c  iparent   = binary array of chromosomes of the parents
c  istart    = the generation to be started from
c  jbest     = the member in the population with the best fitness
c  jelite    = a counter which tracks the number of bits of an individual
c              which match those of the best individual
c  jend      = used in conjunction with iend for debugging
c  jstart    = used in conjunction with iskip for debugging
c  kount     = a counter which controls how frequently the restart
c              file is written
c  kelite    = kelite set to unity when jelite=nchrome, indicates that
c              the best parent was replicated amongst the children
c  mate1     = the number of the population member chosen as mate1
c  mate2     = the number of the population member chosen as mate2
c  nchrmax   = maximum # of chromosomes (binary bits) per individual
c  nchrome   = number of chromosomes (binary bits) of each individual
c  ncreep    = # of creep mutations which occurred during reproduction
c  nmutate   = # of jump mutations which occurred during reproduction
c  nparmax   = maximum # of parameters which the chromosomes make up
c  paramav   = the average of each parameter in the population
c  paramsm   = the sum of each parameter in the population
c  parent    = the floating point parameter array of the parents
c  pardel    = array of the difference between parmax and parmin
c  rand      = the value of the current random number
```

```
c  npossum  = sum of the number of possible values of all parameters
c  tarray   = time array used with 'etime' timing function
c  time0    = clock time at start of run
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c  Subroutines:
c  ____________
c
c  code     = Codes floating point value to binary string.
c  crosovr  = Performs crossover (single-point or uniform).
c  decode   = Decodes binary string to floating point value.
c  evalout  = Evaluates the fitness of each individual and outputs
c             generational information to the 'ga.out' file.
c  func     = The function which is being evaluated.
c  gamicro  = Implements the micro-GA technique.
c  input    = Inputs information from the 'ga.inp' file.
c  initial  = Program initialization and inputs information from the
c             'ga.restart' file.
c  mutate   = Performs mutation (jump and/or creep).
c  newgen   = Writes child array back into parent array for new
c             generation; also checks to see if best individual was
c             replicated (elitism).
c  niche    = Performs niching (sharing) on population.
c  possibl  = Checks to see if decoded binary string falls within
c             specified range of parmin and parmax.
c  ran3     = The random number generator.
c  restart  = Writes the 'ga.restart' file.
c  select   = A subroutine of 'selectn'.
c  selectn  = Performs selection; tournament selection is the only
c             option in this version of the code.
c  shuffle  = Shuffles the population randomly for selection.
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       call etime(tarray)
c       write(6,*) tarray(1),tarray(2)
c       cpu0=tarray(1)
c
c  Call the input subroutine.
c       TIME0=SECNDS(0.0)
      call input
c
c  Perform necessary initialization and read the ga.restart file.
      call initial(istart,npossum,ig2sum)
c
c  $$$$$ Main generational processing loop. $$$$$
      kount=0
      do 20 i=istart,maxgen+istart-1
         write (6,1111) i
         write (24,1111) i
         write(24,1050)
c
c  Evaluate the population, assign fitness, establish the best
c  individual, and write output information.
         call evalout(iskip,iend,ibest,fbar,best)
         geni(i)=float(i)
         genavg(i)=fbar
         genmax(i)=best
         if(npopsiz.eq.1 .or. iskip.ne.0) then
            close(24)
            stop
         endif
c
```

```
c  Implement "niching".
         if (iniche.ne.0) call niche
c
c  Enter selection, crossover and mutation loop.
         ncross=0
         ipick=npopsiz
         do 45 j=1,npopsiz,nchild
c
c  Perform selection.
            call selectn(ipick,j,mate1,mate2)
c
c  Now perform crossover between the randomly selected pair.
            call crosovr(ncross,j,mate1,mate2)
 45      continue
         write(6,1225) ncross
         write(24,1225) ncross
c
c  Now perform random mutations.  If running micro-GA, skip mutation.
         if (microga.eq.0) call mutate
c
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated.
         call newgen(ielite,npossum,ig2sum,ibest)
c
c  Implement micro-GA if enabled.
         if (microga.ne.0) call gamicro(i,npossum,ig2sum,ibest)
c
c  Write to restart file.
         call restart(i,istart,kount)
 20   continue
c  $$$$$ End of main generational processing loop. $$$$$
c 999  continue
      write(24,3000)
      do 100 i=istart,maxgen+istart-1
         evals=float(npopsiz)*geni(i)
         write(24,3100) geni(i),evals,genavg(i),genmax(i)
 100  continue
c      call etime(tarray)
c      write(6,*) tarray(1),tarray(2)
c      cpu1=tarray(1)
c      cpu=(cpu1-cpu0)
c      write(6,1400) cpu,cpu/60.0
c      write(24,1400) cpu,cpu/60.0
      CLOSE (24)
c
 1050 format(1x,' #      Binary Code',16x,'Param1  Param2  Fitness')
 1111 format(//'#################  Generation',i5,'  #################')
 1225 format(/'  Number of Crossovers      =',i5)
c 1400 format(2x,'CPU time for all generations=',e12.6,' sec'/
c     +       2x,'                             ',e12.6,' min')
 3000 format(2x//'Summary of Output'/
     +       2x,'Generation   Evaluations   Avg.Fitness   Best Fitness')
 3100 format(2x,3(e10.4,4x),e11.5)
c
      stop
      end
c
c#####################################################################
      subroutine input
c
c  This subroutine inputs information from the ga.inp (gafort.in) file.
c
      implicit real*8 (a-h,o-z)
      save
```

```
c
      include 'params.f'
      dimension nposibl(nparmax),nichflg(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
      namelist / ga   / irestrt,npopsiz,pmutate,maxgen,idum,pcross,
     +                  itourny,ielite,icreep,pcreep,iunifrm,iniche,
     +                  iskip,iend,nchild,nparam,parmin,parmax,nposibl,
     +                  nowrite,nichflg,microga,kountmx
c
      kountmx=5
      irestrt=0
      itourny=0
      ielite=0
      iunifrm=0
      iniche=0
      iskip=0
      iend=0
      nchild=1
      do 2 i=1,nparam
         nichflg(i)=1
 2    continue
      microga=0
c
      OPEN (UNIT=24, FILE='ga.out', STATUS='UNKNOWN')
      rewind 24
      OPEN (UNIT=23, FILE='ga.inp', STATUS='OLD')
      READ (23, NML = ga)
      CLOSE (23)
      itourny=1
c      if (itourny.eq.0) nchild=2
c
c  Check for array sizing errors.
      if (npopsiz.gt.indmax) then
         write(6,1600) npopsiz
         write(24,1600) npopsiz
         close(24)
         stop
      endif
      if (nparam.gt.nparmax) then
         write(6,1700) nparam
         write(24,1700) nparam
         close(24)
         stop
      endif
c
c  If using the microga option, reset some input variables
      if (microga.ne.0) then
         pmutate=0.0d0
         pcreep=0.0d0
         itourny=1
         ielite=1
         iniche=0
         nchild=1
         if (iunifrm.eq.0) then
            pcross=1.0d0
```

```
         else
            pcross=0.5d0
         endif
      endif
c
 1600 format(1x,'ERROR: npopsiz > indmax.  Set indmax = ',i6)
 1700 format(1x,'ERROR: nparam > nparmax.  Set nparmax = ',i6)
c
      return
      end
c
c#####################################################################
      subroutine initial(istart,npossum,ig2sum)
c
c  This subroutine sets up the program by generating the g0, g1 and
c  ig2 arrays, and counting the number of chromosomes required for the
c  specified input.  The subroutine also initializes the random number
c  generator, parent and iparent arrays (reads the ga.restart file).
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension nposibl(nparmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c
      do 3 i=1,nparam
         g0(i)=parmin(i)
         pardel(i)=parmax(i)-parmin(i)
         g1(i)=pardel(i)/dble(nposibl(i)-1)
 3    continue
      do 6 i=1,nparam
         do 7 j=1,30
            n2j=2**j
            if (n2j.ge.nposibl(i)) then
               ig2(i)=j
               goto 8
            endif
            if (j.ge.30) then
               write(6,2000)
               write(24,2000)
               close(24)
               stop
            endif
 7       continue
 8       continue
 6    continue
c
c  Count the total number of chromosomes (bits) required
      nchrome=0
      npossum=0
      ig2sum=0
      do 9 i=1,nparam
```

```
         nchrome=nchrome+ig2(i)
         npossum=npossum+nposibl(i)
         ig2sum=ig2sum+(2**ig2(i))
 9    continue
      if (nchrome.gt.nchrmax) then
         write(6,1800) nchrome
         write(24,1800) nchrome
         close(24)
         stop
      endif
c
      if (npossum.lt.ig2sum .and. microga.ne.0) then
         write(6,2100)
         write(24,2100)
      endif
c
c  Initialize random number generator
      call ran3(idum,rand)
c
      if(irestrt.eq.0) then
c  Initialize the random distribution of parameters in the individual
c  parents when irestrt=0.
         istart=1
         do 10 i=1,npopsiz
            do 15 j=1,nchrome
               call ran3(1,rand)
               iparent(j,i)=1
               if(rand.lt.0.5d0) iparent(j,i)=0
 15         continue
 10      continue
         if (npossum.lt.ig2sum) call possibl(parent,iparent)
      else
c  If irestrt.ne.0, read from restart file.
         OPEN (UNIT=25, FILE='ga.restart', STATUS='OLD')
         rewind 25
         read(25,*) istart,npopsiz
         do 1 j=1,npopsiz
            read(25,*) k,(iparent(l,j),l=1,nchrome)
 1       continue
         CLOSE (25)
      endif
c
      if(irestrt.ne.0) call ran3(idum-istart,rand)
c
 1800 format(1x,'ERROR: nchrome > nchrmax.  Set nchrmax = ',i6)
 2000 format(1x,'ERROR: You have a parameter with a number of '/
     +       1x,'   possibilities > 2**30!  If you really desire this,'/
     +       1x,'   change the DO loop 7 statement and recompile.'//
     +       1x,'   You may also need to alter the code to work with'/
     +       1x,'   REAL numbers rather than INTEGER numbers; Fortran'/
     +       1x,'   does not like to compute 2**j when j>30.')
 2100 format(1x,'WARNING: for some cases, a considerable performance'/
     +       1x,'   reduction has been observed when running a non-'/
     +       1x,'   optimal number of bits with the micro-GA.'/
     +       1x,'   If possible, use values for nposibl of 2**n,'/
     +       1x,'   e.g. 2, 4, 8, 16, 32, 64, etc.  See ReadMe file.')
c
      return
      end
c
c#######################################################################
      subroutine evalout(iskip,iend,ibest,fbar,best)
c
c  This subroutine evaluates the population, assigns fitness,
```

```
c  establishes the best individual, and outputs information.
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
      dimension paramsm(nparmax),paramav(nparmax),ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
c
      fitsum=0.0d0
      best=-1.0d10
      do 29 n=1,nparam
         paramsm(n)=0.0d0
 29   continue
      jstart=1
      jend=npopsiz
      if(iskip.ne.0) jstart=iskip
      if(iend.ne.0) jend=iend
      do 30 j=jstart,jend
         call decode(j,parent,iparent)
         if(iskip.ne.0 .and. iend.ne.0 .and. iskip.eq.iend)
     +   write(6,1075) j,(iparent(k,j),k=1,nchrome),
     +                   (parent(kk,j),kk=1,nparam),0.0
c
c  Call function evaluator, write out individual and fitness, and add
c  to the summation for later averaging.
         call func(j,funcval)
         fitness(j)=funcval
         write(24,1075) j,(iparent(k,j),k=1,nchrome),
     +                  (parent(kk,j),kk=1,nparam),fitness(j)
         fitsum=fitsum+fitness(j)
         do 22 n=1,nparam
            paramsm(n)=paramsm(n)+parent(n,j)
 22      continue
c
c  Check to see if fitness of individual j is the best fitness.
         if (fitness(j).gt.best) then
            best=fitness(j)
            jbest=j
            do 24 k=1,nchrome
               ibest(k)=iparent(k,j)
 24         continue
         endif
 30   continue
c
c  Compute parameter and fitness averages.
      fbar=fitsum/dble(npopsiz)
      do 23 n=1,nparam
         paramav(n)=paramsm(n)/dble(npopsiz)
 23   continue
c
c  Write output information
      if (npopsiz.eq.1) then
         write(24,1075) 1,(iparent(k,1),k=1,nchrome),
     +                  (parent(k,1),k=1,nparam),fitness(1)
         write(24,*) ' Average Values:'
         write(24,1275) (parent(k,1),k=1,nparam),fbar
      else
         write(24,1275) (paramav(k),k=1,nparam),fbar
```

```
      endif
      write(6,1100) fbar
      write(24,1100) fbar
      write(6,1200) best
      write(24,1200) best
c
 1075 format(i3,1x,30i1,2(1x,f7.4),1x,f8.5)
 1100 format(1x,'Average Function Value of Generation=',f8.5)
 1200 format(1x,'Maximum Function Value              =',f8.5/)
 1275 format(/' Average Values:',18x,2(1x,f7.4),1x,f8.5/)
      return
      end
c
c#######################################################################
      subroutine niche
c
c  Implement "niching" through Goldberg's multidimensional phenotypic
c  sharing scheme with a triangular sharing function.  To find the
c  multidimensional distance from the best individual, normalize all
c  parameter differences.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax),nposibl(nparmax),nichflg(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga8   / nichflg
c
c   Variable definitions:
c
c  alpha   = power law exponent for sharing function; typically = 1.0
c  del     = normalized multidimensional distance between ii and all
c            other members of the population
c            (equals the square root of del2)
c  del2    = sum of the squares of the normalized multidimensional
c            distance between member ii and all other members of
c            the population
c  nniche  = number of niched parameters
c  sigshar = normalized distance to be compared with del; in some sense,
c            1/sigshar can be viewed as the number of regions over which
c            the sharing function should focus, e.g. with sigshar=0.1,
c            the sharing function will try to clump in ten distinct
c            regions of the phase space.  A value of sigshar on the
c            order of 0.1 seems to work best.
c  share   = sharing function between individual ii and j
c  sumshar = sum of the sharing functions for individual ii
c
c       alpha=1.0
      sigshar=0.1d0
      nniche=0
      do 33 jj=1,nparam
         nniche=nniche+nichflg(jj)
 33   continue
      if (nniche.eq.0) then
         write(6,1900)
         write(24,1900)
```

```
            close(24)
            stop
         endif
         do 34 ii=1,npopsiz
            sumshar=0.0d0
            do 35 j=1,npopsiz
               del2=0.0d0
               do 36 k=1,nparam
                  if (nichflg(k).ne.0) then
                     del2=del2+((parent(k,j)-parent(k,ii))/pardel(k))**2
                  endif
 36            continue
               del=(dsqrt(del2))/dble(nniche)
               if (del.lt.sigshar) then
c                   share=1.0-((del/sigshar)**alpha)
                  share=1.0d0-(del/sigshar)
               else
                  share=0.0d0
               endif
               sumshar=sumshar+share/dble(npopsiz)
 35         continue
            if (sumshar.ne.0.0d0) fitness(ii)=fitness(ii)/sumshar
 34      continue
c
 1900 format(1x,'ERROR: iniche=1 and all values in nichflg array = 0'/
     +       1x,'       Do you want to niche or not?')
c
      return
      end
c
c#######################################################################
      subroutine selectn(ipick,j,mate1,mate2)
c
c  Subroutine for selection operator.  Presently, tournament selection
c  is the only option available.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension fitness(indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c  If tournament selection is chosen (i.e. itourny=1), then
c  implement "tournament" selection for selection of new population.
      if(itourny.eq.1) then
         call select(mate1,ipick)
         call select(mate2,ipick)
c        write(3,*) mate1,mate2,fitness(mate1),fitness(mate2)
         do 46 n=1,nchrome
            ichild(n,j)=iparent(n,mate1)
            if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate2)
 46      continue
      endif
```

```
c
      return
      end
c
c#######################################################################
      subroutine crosovr(ncross,j,mate1,mate2)
c
c  Subroutine for crossover between the randomly selected pair.
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
c
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
      if (iunifrm.eq.0) then
c  Single-point crossover at a random chromosome point.
         call ran3(1,rand)
         if(rand.gt.pcross) goto 69
         ncross=ncross+1
         call ran3(1,rand)
         icross=2+dint(dble(nchrome-1)*rand)
         do 50 n=icross,nchrome
            ichild(n,j)=iparent(n,mate2)
            if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
 50      continue
      else
c  Perform uniform crossover between the randomly selected pair.
         do 60 n=1,nchrome
            call ran3(1,rand)
            if(rand.le.pcross) then
               ncross=ncross+1
               ichild(n,j)=iparent(n,mate2)
               if(nchild.eq.2) ichild(n,j+1)=iparent(n,mate1)
            endif
 60      continue
      endif
 69   continue
c
      return
      end
c
c#######################################################################
      subroutine mutate
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension nposibl(nparmax)
      dimension child(nparmax,indmax),ichild(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
```

```
      common / ga6   / parmax,parmin,pardel,nposibl
      common / ga7   / child,ichild
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx
c
c  This subroutine performs mutations on the children generation.
c  Perform random jump mutation if a random number is less than pmutate.
c  Perform random creep mutation if a different random number is less
c  than pcreep.
      nmutate=0
      ncreep=0
      do 70 j=1,npopsiz
         do 75 k=1,nchrome
c  Jump mutation
            call ran3(1,rand)
            if (rand.le.pmutate) then
               nmutate=nmutate+1
               if(ichild(k,j).eq.0) then
                  ichild(k,j)=1
               else
                  ichild(k,j)=0
               endif
               if (nowrite.eq.0) write(6,1300) j,k
               if (nowrite.eq.0) write(24,1300) j,k
            endif
 75      continue
c  Creep mutation (one discrete position away).
         if (icreep.ne.0) then
            do 76 k=1,nparam
               call ran3(1,rand)
               if(rand.le.pcreep) then
                  call decode(j,child,ichild)
                  ncreep=ncreep+1
                  creep=1.0d0
                  call ran3(1,rand)
                  if (rand.lt.0.5d0) creep=-1.0d0
                  child(k,j)=child(k,j)+g1(k)*creep
                  if (child(k,j).gt.parmax(k)) then
                     child(k,j)=parmax(k)-1.0d0*g1(k)
                  elseif (child(k,j).lt.parmin(k)) then
                     child(k,j)=parmin(k)+1.0d0*g1(k)
                  endif
                  call code(j,k,child,ichild)
                  if (nowrite.eq.0) write(6,1350) j,k
                  if (nowrite.eq.0) write(24,1350) j,k
               endif
 76         continue
         endif
 70   continue
      write(6,1250) nmutate,ncreep
      write(24,1250) nmutate,ncreep
c
 1250 format(/'  Number of Jump Mutations  =',i5/
     +        '  Number of Creep Mutations =',i5)
 1300 format('*** Jump mutation performed on individual  ',i4,
     +       ', chromosome ',i3,' ***')
 1350 format('*** Creep mutation performed on individual ',i4,
     +       ', parameter  ',i3,' ***')
c
      return
      end
c
c#######################################################################
```

```
      subroutine newgen(ielite,npossum,ig2sum,ibest)
c
c  Write child array back into parent array for new generation.  Check
c  to see if the best parent was replicated; if not, and if ielite=1,
c  then reproduce the best parent into a random slot.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),child(nparmax,indmax)
      dimension iparent(nchrmax,indmax),ichild(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga7   / child,ichild
c
      if (npossum.lt.ig2sum) call possibl(child,ichild)
      kelite=0
      do 94 j=1,npopsiz
         jelite=0
         do 95 n=1,nchrome
            iparent(n,j)=ichild(n,j)
            if (iparent(n,j).eq.ibest(n)) jelite=jelite+1
            if (jelite.eq.nchrome) kelite=1
 95      continue
 94   continue
      if (ielite.ne.0 .and. kelite.eq.0) then
         call ran3(1,rand)
         irand=1d0+dint(dble(npopsiz)*rand)
         do 96 n=1,nchrome
            iparent(n,irand)=ibest(n)
 96      continue
         write(24,1260) irand
      endif
c
 1260 format('  Elitist Reproduction on Individual ',i4)
c
      return
      end
c
c#######################################################################
      subroutine gamicro(i,npossum,ig2sum,ibest)
c
c  Micro-GA implementation subroutine
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension ibest(nchrmax)
c
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
c
c  First, check for convergence of micro population.
c  If converged, start a new generation with best individual and fill
c  the remainder of the population with new randomly generated parents.
c
c  Count number of different bits from best member in micro-population
```

```
      icount=0
      do 81 j=1,npopsiz
         do 82 n=1,nchrome
            if(iparent(n,j).ne.ibest(n)) icount=icount+1
 82      continue
 81   continue
c
c  If icount less than 5% of number of bits, then consider population
c  to be converged.  Restart with best individual and random others.
      diffrac=dble(icount)/dble((npopsiz-1)*nchrome)
      if (diffrac.lt.0.05d0) then
      do 87 n=1,nchrome
         iparent(n,1)=ibest(n)
 87   continue
      do 88 j=2,npopsiz
         do 89 n=1,nchrome
            call ran3(1,rand)
            iparent(n,j)=1
            if(rand.lt.0.5d0) iparent(n,j)=0
 89      continue
 88   continue
      if (npossum.lt.ig2sum) call possibl(parent,iparent)
      write(6,1375) i
      write(24,1375) i
      endif
c
 1375 format(//'%%%%%%%  Restart micro-population at generation',
     +       i5,'  %%%%%%%')
c
      return
      end
c
c#######################################################################
      subroutine select(mate,ipick)
c
c  This routine selects the better of two possible parents for mating.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
c
      if(ipick+1.gt.npopsiz) call shuffle(ipick)
      ifirst=ipick
      isecond=ipick+1
      ipick=ipick+2
      if(fitness(ifirst).gt.fitness(isecond)) then
         mate=ifirst
      else
         mate=isecond
      endif
c     write(3,*)'select',ifirst,isecond,fitness(ifirst),fitness(isecond)
c
      return
      end
c
c#######################################################################
      subroutine shuffle(ipick)
```

```
c
c  This routine shuffles the parent array and its corresponding fitness
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      common / ga4   / fitness
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      dimension fitness(indmax)
c
      ipick=1
      do 10 j=1,npopsiz-1
         call ran3(1,rand)
         iother=j+1+dint(dble(npopsiz-j)*rand)
         do 20 n=1,nchrome
            itemp=iparent(n,iother)
            iparent(n,iother)=iparent(n,j)
            iparent(n,j)=itemp
 20      continue
         temp=fitness(iother)
         fitness(iother)=fitness(j)
         fitness(j)=temp
 10   continue
c
      return
      end
c
c#######################################################################
      subroutine decode(i,array,iarray)
c
c  This routine decodes a binary string to a real number.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
c
      l=1
      do 10 k=1,nparam
         iparam=0
         m=l
         do 20 j=m,m+ig2(k)-1
            l=l+1
            iparam=iparam+iarray(j,i)*(2**(m+ig2(k)-1-j))
 20      continue
         array(k,i)=g0(k)+g1(k)*dble(iparam)
 10   continue
c
      return
      end
c
c#######################################################################
      subroutine code(j,k,array,iarray)
c
c  This routine codes a parameter into a binary string.
c
```

```
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax)
c
c  First, establish the beginning location of the parameter string of
c  interest.
      istart=1
      do 10 i=1,k-1
         istart=istart+ig2(i)
 10   continue
c
c  Find the equivalent coded parameter value, and back out the binary
c  string by factors of two.
      m=ig2(k)-1
      if (g1(k).eq.0.0d0) return
      iparam=nint((array(k,j)-g0(k))/g1(k))
      do 20 i=istart,istart+ig2(k)-1
         iarray(i,j)=0
         if ((iparam+1).gt.(2**m)) then
            iarray(i,j)=1
            iparam=iparam-2**m
         endif
         m=m-1
 20   continue
c     write(3,*)array(k,j),iparam,(iarray(i,j),i=istart,istart+ig2(k)-1)
c
      return
      end
c
c#######################################################################
c
      subroutine possibl(array,iarray)
c
c  This subroutine determines whether or not all parameters are within
c  the specified range of possibility.  If not, the parameter is
c  randomly reassigned within the range.  This subroutine is only
c  necessary when the number of possibilities per parameter is not
c  optimized to be 2**n, i.e. if npossum < ig2sum.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga5   / g0,g1,ig2
      common / ga6   / parmax,parmin,pardel,nposibl
      dimension array(nparmax,indmax),iarray(nchrmax,indmax)
      dimension g0(nparmax),g1(nparmax),ig2(nparmax),nposibl(nparmax)
      dimension parmax(nparmax),parmin(nparmax),pardel(nparmax)
c
      do 10 i=1,npopsiz
         call decode(i,array,iarray)
         do 20 j=1,nparam
            n2ig2j=2**ig2(j)
            if(nposibl(j).ne.n2ig2j .and. array(j,i).gt.parmax(j)) then
               call ran3(1,rand)
               irand=dint(dble(nposibl(j))*rand)
               array(j,i)=g0(j)+dble(irand)*g1(j)
```

```
                call code(i,j,array,iarray)
                if (nowrite.eq.0) write(6,1000) i,j
                if (nowrite.eq.0) write(24,1000) i,j
             endif
 20       continue
 10    continue
c
 1000 format('*** Parameter adjustment to individual     ',i4,
     +       ', parameter  ',i3,' ***')
c
      return
      end
c
c#######################################################################
      subroutine restart(i,istart,kount)
c
c  This subroutine writes restart information to the ga.restart file.
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      common / ga1   / npopsiz,nowrite
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
      dimension parent(nparmax,indmax),iparent(nchrmax,indmax)
      common /inputga/ pcross,pmutate,pcreep,maxgen,idum,irestrt,
     +                 itourny,ielite,icreep,iunifrm,iniche,
     +                 iskip,iend,nchild,microga,kountmx

      kount=kount+1
      if(i.eq.maxgen+istart-1 .or. kount.eq.kountmx) then
         OPEN (UNIT=25, FILE='ga.restart', STATUS='OLD')
         rewind 25
         write(25,*) i+1,npopsiz
         do 80 j=1,npopsiz
            write(25,1500) j,(iparent(l,j),l=1,nchrome)
 80      continue
         CLOSE (25)
         kount=0
      endif
c
 1500 format(i5,3x,30i2)
c
      return
      end
c
c#######################################################################
      subroutine ran3(idum,rand)
c
c  Returns a uniform random deviate between 0.0 and 1.0.  Set idum to
c  any negative value to initialize or reinitialize the sequence.
c  This function is taken from W.H. Press', "Numerical Recipes" p. 199.
c
      implicit real*8 (a-h,m,o-z)
      save
c       implicit real*4(m)
      parameter (mbig=4000000.,mseed=1618033.,mz=0.,fac=1./mbig)
c     parameter (mbig=1000000000,mseed=161803398,mz=0,fac=1./mbig)
c
c  According to Knuth, any large mbig, and any smaller (but still large)
c  mseed can be substituted for the above values.
      dimension ma(55)
      data iff /0/
```

```
      if (idum.lt.0 .or. iff.eq.0) then
         iff=1
         mj=mseed-dble(iabs(idum))
         mj=dmod(mj,mbig)
         ma(55)=mj
         mk=1
         do 11 i=1,54
            ii=mod(21*i,55)
            ma(ii)=mk
            mk=mj-mk
            if(mk.lt.mz) mk=mk+mbig
            mj=ma(ii)
 11      continue
         do 13 k=1,4
            do 12 i=1,55
               ma(i)=ma(i)-ma(1+mod(i+30,55))
               if(ma(i).lt.mz) ma(i)=ma(i)+mbig
 12         continue
 13      continue
         inext=0
         inextp=31
         idum=1
      endif
      inext=inext+1
      if(inext.eq.56) inext=1
      inextp=inextp+1
      if(inextp.eq.56) inextp=1
      mj=ma(inext)-ma(inextp)
      if(mj.lt.mz) mj=mj+mbig
      ma(inext)=mj
      rand=mj*fac
      return
      end
c
c#########################################################################
c
      subroutine func(j,funcval)
c
      implicit real*8 (a-h,o-z)
      save
c
      include 'params.f'
      dimension parent(nparmax,indmax)
      dimension iparent(nchrmax,indmax)
c      dimension parent2(indmax,nparmax),iparent2(indmax,nchrmax)
c
      common / ga2   / nparam,nchrome
      common / ga3   / parent,iparent
c
c  This is an N-dimensional version of the multimodal function with
c  decreasing peaks used by Goldberg and Richardson (1987, see ReadMe
c  file for complete reference).  In N dimensions, this function has
c  (nvalley-1)^nparam peaks, but only one global maximum.  It is a
c  reasonably tough problem for the GA, especially for higher dimensions
c  and larger values of nvalley.
c
      nvalley=6
      pi=4.0d0*datan(1.d0)
      funcval=1.0d0
      do 10 i=1,nparam
         f1=(sin(5.1d0*pi*parent(i,j) + 0.5d0))**nvalley
         f2=exp(-4.0d0*log(2.0d0)*((parent(i,j)-0.0667d0)**2)/0.64d0)
         funcval=funcval*f1*f2
 10   continue
```

```
c
c  As mentioned in the ReadMe file, The arrays have been rearranged
c  to enable a more efficient caching of system memory.  If this causes
c  interface problems with existing functions used with previous
c  versions of my code, then you can use some temporary arrays to bridge
c  this version with older versions.  I've named the temporary arrays
c  parent2 and iparent2.  If you want to use these arrays, uncomment the
c  dimension statement above as well as the following do loop lines.
c
c       do 11 i=1,nparam
c          parent2(j,i)=parent(i,j)
c 11    continue
c       do 12 k=1,nchrome
c          iparent2(j,k)=iparent(k,j)
c 12    continue
c
      return
      end
c#########################################################################
```

```
$ga
irestrt=0,
microga=1,
npopsiz= 5,
nparam= 2,
pmutate=0.02d0,
maxgen=200,
idum=-1000,
pcross=0.5d0,
itourny=1,
ielite=1,
icreep=0,
pcreep=0.04d0,
iunifrm=1,
iniche=0,
nchild=1,
iskip= 0, iend=  0,
nowrite=1,
kountmx=5,
parmin= 2*0.0d0,
parmax= 2*1.0d0,
nposibl=2*32768,
nichflg=2*1,
$end
```

```
&ga
irestrt=0,
microga=1,
npopsiz= 5,
nparam= 2,
pmutate=0.02d0,
maxgen=200,
idum=-1000,
pcross=0.5d0,
itourny=1,
ielite=1,
icreep=0,
pcreep=0.04d0,
iunifrm=1,
iniche=0,
nchild=1,
iskip= 0, iend=  0,
nowrite=1,
kountmx=5,
parmin= 2*0.0d0,
parmax= 2*1.0d0,
nposibl=2*32768,
nichflg=2*1,
/
```

```
#################  Generation    1  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 100011111101010001001000001101  0.5618  0.1410  0.00000
  2 000110010010001101001110011100  0.0982  0.6532  0.10147
  3 110010101011001110110101011010  0.7918  0.8543  0.00027
  4 010100111000101110111111010001  0.3263  0.8736  0.00064
  5 111011111100000100100111110101  0.9366  0.5778  0.00000

 Average Values:                      0.5429  0.6200  0.02047

 Average Function Value of Generation= 0.02047
 Maximum Function Value              = 0.10147


  Number of Crossovers       =   64
  Elitist Reproduction on Individual    2


#################  Generation    2  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 010010110011001110011111011110  0.2937  0.8115  0.00916
  2 000110010010001101001110011100  0.0982  0.6532  0.10147
  3 110010010010001110111101011000  0.7857  0.8699  0.00008
  4 000100011010101101011110011101  0.0690  0.6845  0.09532
  5 110110101011101110110111011011  0.8544  0.8583  0.00430

 Average Values:                      0.4202  0.7755  0.04206

 Average Function Value of Generation= 0.04206
 Maximum Function Value              = 0.10147


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    4


#################  Generation    3  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 000010010011001110001110011100  0.0359  0.7782  0.00019
  2 000100011010101101001110011100  0.0690  0.6532  0.22390
  3 000100010010001101001110011100  0.0669  0.6532  0.22471
  4 000110010010001101001110011100  0.0982  0.6532  0.10147
  5 000110010010101101001110011101  0.0983  0.6532  0.10080

 Average Values:                      0.0737  0.6782  0.13021

 Average Function Value of Generation= 0.13021
 Maximum Function Value              = 0.22471


  Number of Crossovers       =   70


%%%%%%%  Restart micro-population at generation    3  %%%%%%%


#################  Generation    4  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 000100010010001101001110011100  0.0669  0.6532  0.22471
  2 101101001010101011000000000110  0.7057  0.3752  0.00000
  3 111001101100010101101000100001  0.9015  0.7042  0.00011
  4 111010011010001010101101010111  0.9127  0.3386  0.00000
  5 010010010010000110111110010000  0.2857  0.8716  0.02351
```

```
Average Values:                     0.5745  0.5886  0.04967

Average Function Value of Generation= 0.04967
Maximum Function Value              = 0.22471


 Number of Crossovers       =   67
 Elitist Reproduction on Individual    5


################  Generation    5  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 110010011010001010111111100 10011 0.7877  0.3717  0.00000
 2 010000010010001111001110010100 0.2544  0.9030  0.00370
 3 000000010010001101111110010100 0.0044  0.7467  0.00000
 4 011100010010001100001100011111 0.4419  0.5244  0.00271
 5 000100010010001101001110011100 0.0669  0.6532  0.22471

Average Values:                     0.3111  0.6398  0.04622

Average Function Value of Generation= 0.04622
Maximum Function Value              = 0.22471


 Number of Crossovers       =   81
 Elitist Reproduction on Individual    2


################  Generation    6  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 110110010010001110101110010101 0.8482  0.8405  0.00482
 2 000100010010001101001110011100 0.0669  0.6532  0.22471
 3 011100010010001101001110011110 0.4419  0.6533  0.09732
 4 010000010010001111001110011101 0.2544  0.9033  0.00358
 5 010100010010001100001100011100 0.3169  0.5243  0.00036

Average Values:                     0.3857  0.7149  0.06616

Average Function Value of Generation= 0.06616
Maximum Function Value              = 0.22471


 Number of Crossovers       =   78
 Elitist Reproduction on Individual    5


################  Generation    7  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 110100010010001111001110010100 0.8169  0.9030  0.00015
 2 011100010010001101001110011110 0.4419  0.6533  0.09732
 3 010100010010001100001110010111 0.3169  0.5281  0.00018
 4 001100010010001101001110011100 0.1919  0.6532  0.00115
 5 000100010010001101001110011100 0.0669  0.6532  0.22471

Average Values:                     0.3669  0.6782  0.06470

Average Function Value of Generation= 0.06470
Maximum Function Value              = 0.22471


 Number of Crossovers       =   80
```

```
#################  Generation    8  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010010001101001110011100  0.0669  0.6532  0.22471
  2 011100010010001101001110011110  0.4419  0.6533  0.09732
  3 010100010010001101001110011110  0.3169  0.6533  0.01271
  4 011100010010001100001110011111  0.4419  0.5283  0.00132
  5 001100010010001101001110011100  0.1919  0.6532  0.00115

 Average Values:                    0.2919  0.6283  0.06744

 Average Function Value of Generation= 0.06744
 Maximum Function Value              = 0.22471


  Number of Crossovers      =   78
  Elitist Reproduction on Individual    5


#################  Generation    9  #################
  #       Binary Code                Param1  Param2  Fitness
  1 011100010010001101001110011110  0.4419  0.6533  0.09732
  2 010100010010001101001110011110  0.3169  0.6533  0.01271
  3 000100010010001100001110011100  0.0669  0.5282  0.00310
  4 011100010010001100001110011111  0.4419  0.5283  0.00132
  5 000100010010001101001110011100  0.0669  0.6532  0.22471

 Average Values:                    0.2669  0.6033  0.06783

 Average Function Value of Generation= 0.06783
 Maximum Function Value              = 0.22471


  Number of Crossovers      =   80

 %%%%%%%  Restart micro-population at generation    9  %%%%%%%

#################  Generation   10  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010010001101001110011100  0.0669  0.6532  0.22471
  2 111001101110111010110001111100  0.9021  0.3475  0.00000
  3 101100010011010100000110101101  0.6922  0.5131  0.00183
  4 010110010110000100101100110011  0.3491  0.5875  0.00000
  5 010001011100011100010111110110  0.2726  0.5466  0.00001

 Average Values:                    0.4566  0.5296  0.04531

 Average Function Value of Generation= 0.04531
 Maximum Function Value              = 0.22471


  Number of Crossovers      =   71
  Elitist Reproduction on Individual    5


#################  Generation   11  #################
  #       Binary Code                Param1  Param2  Fitness
  1 100000001010001001001000111100  0.5025  0.1425  0.00016
  2 010100010000001100011110111110  0.3164  0.5605  0.00000
  3 001100010010000101001110101100  0.1919  0.6537  0.00115
  4 111000010110010000100010101100  0.8805  0.0678  0.02852
  5 000100010010001101001110011100  0.0669  0.6532  0.22471
```

```
 Average Values:                        0.3916  0.4155  0.05091

 Average Function Value of Generation= 0.05091
 Maximum Function Value               = 0.22471


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    1


################# Generation   12  #################
  #       Binary Code                   Param1  Param2  Fitness
  1 000100010010001101001110011100  0.0669  0.6532  0.22471
  2 110100010110001100001110001100  0.8179  0.5277  0.00012
  3 101100010110010001001010101100  0.6930  0.1459  0.00004
  4 001100010010001101001110011100  0.1919  0.6532  0.00115
  5 000100010010000101001110001100  0.0669  0.6527  0.22492

 Average Values:                        0.3673  0.5266  0.09019

 Average Function Value of Generation= 0.09019
 Maximum Function Value               = 0.22492


  Number of Crossovers       =   67


################# Generation   13  #################
  #       Binary Code                   Param1  Param2  Fitness
  1 001100010010000101001110011100  0.1919  0.6532  0.00115
  2 001100010010000101001110011100  0.1919  0.6532  0.00115
  3 000100010010001101001110001100  0.0669  0.6527  0.22492
  4 001100010010001101001110011100  0.1919  0.6532  0.00115
  5 000100010010000101001110001100  0.0669  0.6527  0.22492

 Average Values:                        0.1419  0.6530  0.09066

 Average Function Value of Generation= 0.09066
 Maximum Function Value               = 0.22492


  Number of Crossovers       =   77


%%%%%%%  Restart micro-population at generation   13  %%%%%%%


################# Generation   14  #################
  #       Binary Code                   Param1  Param2  Fitness
  1 000100010010000101001110001100  0.0669  0.6527  0.22492
  2 010001011010000000001101011101  0.2720  0.0263  0.19778
  3 000001101111110101100000111110  0.0273  0.6894  0.01945
  4 101001101011000111110111100100  0.6511  0.9836  0.00012
  5 000010000101100100110111010111  0.0326  0.6081  0.01638

 Average Values:                        0.2100  0.5920  0.09173

 Average Function Value of Generation= 0.09173
 Maximum Function Value               = 0.22492


  Number of Crossovers       =   86
  Elitist Reproduction on Individual    1
```

```
################# Generation   15  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010010000101001110001100   0.0669  0.6527  0.22492
 2 000001111010110000101000011101   0.0300  0.0790  0.29078
 3 000000101111110101101000101110   0.0117  0.7046  0.00132
 4 000000010010000001001111001100   0.0044  0.1547  0.00000
 5 000101010011110101000110111110   0.0829  0.6386  0.16025

 Average Values:                    0.0392  0.4459  0.13545

 Average Function Value of Generation= 0.13545
 Maximum Function Value              = 0.29078


  Number of Crossovers       =   76
  Elitist Reproduction on Individual    3


################# Generation   16  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000101110010110001101000011110   0.0905  0.2040  0.02422
 2 000101010011000101001110001100   0.0828  0.6527  0.18437
 3 000001111010110000101000011101   0.0300  0.0790  0.29078
 4 000001110100100010011000011101   0.0142  0.1488  0.00002
 5 000001110010110100101110011101   0.0280  0.5907  0.00161

 Average Values:                    0.0491  0.3351  0.10020

 Average Function Value of Generation= 0.10020
 Maximum Function Value              = 0.29078


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    2


################# Generation   17  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000001110010110000101000011100   0.0280  0.0790  0.25527
 2 000001111010110000101000011101   0.0300  0.0790  0.29078
 3 000001111010110001101000011111   0.0300  0.2041  0.01239
 4 000101110010110001101000011111   0.0905  0.2041  0.02431
 5 000101110010110000101000011100   0.0905  0.0790  0.57101

 Average Values:                    0.0538  0.1290  0.23075

 Average Function Value of Generation= 0.23075
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   69
  Elitist Reproduction on Individual    3


################# Generation   18  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000001111010110000101000011100   0.0300  0.0790  0.29095
 2 000001110010110000101000011100   0.0280  0.0790  0.25527
 3 000101110010110000101000011100   0.0905  0.0790  0.57101
 4 000101110010110000101000011111   0.0905  0.0791  0.57001
 5 000101110010110000101000011110   0.0905  0.0790  0.57035

 Average Values:                    0.0659  0.0790  0.45152
```

```
 Average Function Value of Generation= 0.45152
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   88


%%%%%%%  Restart micro-population at generation   18  %%%%%%%


#################  Generation   19  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000101110010110000101000011100  0.0905  0.0790  0.57101
  2 000010000011101110101111101100  0.0321  0.8432  0.02611
  3 111100000011100100101111111111  0.9384  0.5937  0.00000
  4 101000000010110000001011110000  0.6257  0.0229  0.02530
  5 100000011010011111001000011011  0.5065  0.8915  0.00085

 Average Values:                    0.4386  0.4861  0.12465

 Average Function Value of Generation= 0.12465
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    3


#################  Generation   20  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000010010011100110101010101100  0.0360  0.8334  0.02844
  2 100100010010110000001001011100  0.5671  0.0184  0.00000
  3 000101110010110000101000011100  0.0905  0.0790  0.57101
  4 100101111010110110001000011000  0.5925  0.7664  0.00000
  5 000011000010101010101101001100  0.0475  0.3383  0.00109

 Average Values:                    0.2667  0.4071  0.12011

 Average Function Value of Generation= 0.12011
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    3


#################  Generation   21  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000010010010100010101000001100  0.0358  0.3285  0.00512
  2 000011110010100110101000101100  0.0592  0.8295  0.05316
  3 000101110010110000101000011100  0.0905  0.0790  0.57101
  4 000001110010110000101010001100  0.0280  0.0824  0.23691
  5 000111100010101010101100001100  0.1178  0.3363  0.00024

 Average Values:                    0.0663  0.3311  0.17329

 Average Function Value of Generation= 0.17329
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    4
```

```
#################  Generation   22  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 00001111001010010010101000011100  0.0592  0.5790  0.00051
  2 00010111001011000010101001 1100   0.0905  0.0829  0.52352
  3 00001111001010001010101000 1100   0.0592  0.3324  0.00522
  4 00010111001011000010100001 1100   0.0905  0.0790  0.57101
  5 00000111001010000010100011 1100   0.0280  0.0800  0.24931

 Average Values:                      0.0655  0.2306  0.26991

 Average Function Value of Generation= 0.26991
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   81
  Elitist Reproduction on Individual    5


#################  Generation   23  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 00010111001010000010100011 1100   0.0905  0.0800  0.56137
  2 00010111001011000010101001 1100   0.0905  0.0829  0.52352
  3 00011111001011000010101001 1100   0.1218  0.0829  0.05388
  4 00000111001011000010100001 1100   0.0280  0.0790  0.25527
  5 00010111001011000010100001 1100   0.0905  0.0790  0.57101

 Average Values:                      0.0843  0.0807  0.39301

 Average Function Value of Generation= 0.39301
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   68

%%%%%%%  Restart micro-population at generation   23  %%%%%%%

#################  Generation   24  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 00010111001011000010100001 1100   0.0905  0.0790  0.57101
  2 11000101001000011001010010 1000   0.7700  0.7903  0.00000
  3 00010110000000011100000101 1101   0.0859  0.8779  0.02459
  4 01011111110101011111100100 1111   0.3743  0.9868  0.00000
  5 01100010011000100000111000 0111   0.3843  0.0276  0.00043

 Average Values:                      0.3410  0.5523  0.11921

 Average Function Value of Generation= 0.11921
 Maximum Function Value              = 0.57101


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    3


#################  Generation   25  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 01010110000000010010010110 01111  0.3360  0.1469  0.00000
  2 00110010001001000010111000 1100   0.1959  0.0902  0.00705
  3 00010111001011000010100001 1100   0.0905  0.0790  0.57101
  4 01110110000000101000100101 0101   0.4610  0.2682  0.41717
  5 00010111001011001110100101 1100   0.0905  0.4559  0.32995
```

```
Average Values:                     0.2348  0.2081  0.26504

Average Function Value of Generation= 0.26504
Maximum Function Value              = 0.57101


 Number of Crossovers       =   63
 Elitist Reproduction on Individual    2


################# Generation   26  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000101110010001010001000011100  0.0904  0.2665  0.53676
  2 000101110010110000101000011100  0.0905  0.0790  0.57101
  3 011101110010001000101000010100  0.4654  0.0787  0.43583
  4 000101110010110000101010001100  0.0905  0.0824  0.52995
  5 011101100010100011001001011101  0.4616  0.3935  0.00485

Average Values:                     0.2397  0.1800  0.41568

Average Function Value of Generation= 0.41568
Maximum Function Value              = 0.57101


 Number of Crossovers       =   79


################# Generation   27  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000101110010101010101000011100  0.0905  0.3290  0.00659
  2 000101110010011000101010001100  0.0904  0.0824  0.53182
  3 000101110010110000101000011100  0.0905  0.0790  0.57101
  4 000101110010010000101000011100  0.0904  0.0790  0.57370
  5 000101110010110000101000011100  0.0905  0.0790  0.57101

Average Values:                     0.0905  0.1297  0.45083

Average Function Value of Generation= 0.45083
Maximum Function Value              = 0.57370


 Number of Crossovers       =   73


%%%%%%%  Restart micro-population at generation   27  %%%%%%%


################# Generation   28  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000101110010010000101000011100  0.0904  0.0790  0.57370
  2 110101000010001011011100100001  0.8287  0.4307  0.01616
  3 111111110100101000111101001110  0.9973  0.1196  0.00023
  4 110011000101000101010101111011  0.7981  0.6678  0.00150
  5 010011000110110101100011101110  0.2985  0.6948  0.01379

Average Values:                     0.6026  0.3984  0.12107

Average Function Value of Generation= 0.12107
Maximum Function Value              = 0.57370


 Number of Crossovers       =   65
 Elitist Reproduction on Individual    4
```

```
################  Generation   29  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 110101000101000100101001111011  0.8294  0.5819  0.00006
  2 110101110010011000011000111100  0.8404  0.0487  0.05285
  3 010101110110110000101000101110  0.3415  0.0795  0.00053
  4 000101110010010000101000011100  0.0904  0.0790  0.57370
  5 110111000010101101101011101101  0.8600  0.7104  0.00065

 Average Values:                    0.5924  0.2999  0.12556

 Average Function Value of Generation= 0.12556
 Maximum Function Value              = 0.57370


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    2


################  Generation   30  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000101100010010001101001111100  0.0865  0.2069  0.03992
  2 000101110010010000101000011100  0.0904  0.0790  0.57370
  3 100101110010010000001000111100  0.5904  0.0175  0.00064
  4 110101110010001000111001111101  0.8404  0.1132  0.01082
  5 100101110010011000101000011100  0.5904  0.0790  0.00479

 Average Values:                    0.4396  0.0991  0.12597

 Average Function Value of Generation= 0.12597
 Maximum Function Value              = 0.57370


  Number of Crossovers       =   80
  Elitist Reproduction on Individual    5


################  Generation   31  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000101110010010001101001011100  0.0904  0.2059  0.03095
  2 100101110010001001101001111100  0.5904  0.2069  0.00029
  3 010101110010010000111000111101  0.3404  0.1112  0.00015
  4 100101100010010000101001011100  0.5865  0.0809  0.00235
  5 000101110010010000101000011100  0.0904  0.0790  0.57370

 Average Values:                    0.3396  0.1368  0.12149

 Average Function Value of Generation= 0.12149
 Maximum Function Value              = 0.57370


  Number of Crossovers       =   75


################  Generation   32  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000101110010010000101000011100  0.0904  0.0790  0.57370
  2 000101100010010000101000011100  0.0865  0.0790  0.65745
  3 000101110010010000101001011100  0.0904  0.0809  0.55103
  4 100101100010010000101000011100  0.5865  0.0790  0.00245
  5 000101110010010001101001011100  0.0904  0.2059  0.03095

 Average Values:                    0.1888  0.1048  0.36312
```

```
 Average Function Value of Generation= 0.36312
 Maximum Function Value              = 0.65745


  Number of Crossovers       =   66


################# Generation   33  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000101110010010001101000011100  0.0904  0.2040  0.02414
  2 000101100010010000101000011100  0.0865  0.0790  0.65745
  3 000101110010010001101000011100  0.0904  0.2040  0.02414
  4 000101110010010000101000011100  0.0904  0.0790  0.57370
  5 000101110010010000101000011100  0.0904  0.0790  0.57370

 Average Values:                    0.0896  0.1290  0.37062

 Average Function Value of Generation= 0.37062
 Maximum Function Value              = 0.65745


  Number of Crossovers       =   63
  Elitist Reproduction on Individual    5


%%%%%%  Restart micro-population at generation   33  %%%%%%


################# Generation   34  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000101100010010000101000011100  0.0865  0.0790  0.65745
  2 001110111001101110111010100110  0.2328  0.8645  0.02375
  3 111000000101101010101110000110  0.8764  0.3400  0.00003
  4 001000001100010111100100100101  0.1280  0.9465  0.00000
  5 001001110101001010010110011010  0.1536  0.2938  0.00001

 Average Values:                    0.2955  0.5047  0.13625

 Average Function Value of Generation= 0.13625
 Maximum Function Value              = 0.65745


  Number of Crossovers       =   69
  Elitist Reproduction on Individual    1


################# Generation   35  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000101100010010000101000011100  0.0865  0.0790  0.65745
  2 001101110010001000011110011110  0.2154  0.0595  0.12518
  3 001101111101001010010010100110  0.2181  0.2863  0.08669
  4 001001100111011010110110011100  0.1502  0.3563  0.00000
  5 001001100001000000111110011010  0.1487  0.1219  0.00002

 Average Values:                    0.1638  0.1806  0.17387

 Average Function Value of Generation= 0.17387
 Maximum Function Value              = 0.65745


  Number of Crossovers       =   83
  Elitist Reproduction on Individual    5
```

```
################  Generation   36  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 00110111000100100001110011010  0.2151  0.0594  0.12226
 2 00010111110100000011101000011 0  0.0930  0.1135  0.08931
 3 00010111001000100000111001110 0  0.0904  0.0282  0.18690
 4 00110111001000100000111001111 0  0.2154  0.0283  0.03801
 5 00010110001001000010100001110 0  0.0865  0.0790  0.65745

Average Values:                      0.1401  0.0617  0.21879

Average Function Value of Generation= 0.21879
Maximum Function Value              = 0.65745


 Number of Crossovers       =   86
 Elitist Reproduction on Individual    3


################  Generation   37  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 000101101111010000111010010110  0.0897  0.1140  0.09775
 2 000101100001000000101000011110  0.0862  0.0790  0.66297
 3 000101100010010000101000011100  0.0865  0.0790  0.65745
 4 000101110010010000101000011100  0.0904  0.0790  0.57370
 5 000101101111010000111010010110  0.0897  0.1140  0.09775

Average Values:                      0.0885  0.0930  0.41793

Average Function Value of Generation= 0.41793
Maximum Function Value              = 0.66297


 Number of Crossovers       =   75

################  Generation   38  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 000101100001000000101000011110  0.0862  0.0790  0.66297
 2 000101100111010000111010010110  0.0877  0.1140  0.10473
 3 000101100011010000101000011110  0.0867  0.0790  0.65162
 4 000101100010010000101000011110  0.0865  0.0790  0.65668
 5 000101100010000000101000011100  0.0864  0.0790  0.65871

Average Values:                      0.0867  0.0860  0.54695

Average Function Value of Generation= 0.54695
Maximum Function Value              = 0.66297


 Number of Crossovers       =   70

################  Generation   39  ################
 #      Binary Code                 Param1  Param2  Fitness
 1 000101100011000000101000011100  0.0867  0.0790  0.65365
 2 000101100000001000010100001111 0  0.0860  0.0790  0.66672
 3 000101100001000000101000011110  0.0862  0.0790  0.66297
 4 000101100001010000101000011110  0.0862  0.0790  0.66172
 5 000101100010000000101000011110  0.0864  0.0790  0.65795

Average Values:                      0.0863  0.0790  0.66060

Average Function Value of Generation= 0.66060
```

```
 Maximum Function Value             = 0.66672


  Number of Crossovers       =   58


################# Generation   40  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000101100001000000101000011110  0.0862  0.0790  0.66297
  2 000101100001000000101000011110  0.0862  0.0790  0.66297
  3 000101100001010000101000011110  0.0862  0.0790  0.66172
  4 000101100000000000101000011110  0.0859  0.0790  0.66797
  5 000101100000010000101000011110  0.0860  0.0790  0.66672

 Average Values:                     0.0861  0.0790  0.66447

 Average Function Value of Generation= 0.66447
 Maximum Function Value             = 0.66797


  Number of Crossovers       =   76


%%%%%%%  Restart micro-population at generation   40  %%%%%%%


################# Generation   41  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000101100000000000101000011110  0.0859  0.0790  0.66797
  2 100010101111011111000110101100  0.5428  0.8881  0.00000
  3 000110101110110100010001000001  0.1052  0.5332  0.00030
  4 111011110101110111110000010110  0.9350  0.9695  0.00000
  5 011110001100001101101010110111  0.4717  0.7087  0.00561

 Average Values:                     0.4281  0.6357  0.13478

 Average Function Value of Generation= 0.13478
 Maximum Function Value             = 0.66797


  Number of Crossovers       =   69
  Elitist Reproduction on Individual     4


################# Generation   42  #################
  #       Binary Code                Param1  Param2  Fitness
  1 100101100111001011100110001110  0.5877  0.4497  0.00169
  2 000110101000000000101000011111  0.1035  0.0791  0.29375
  3 101010001100001111101010111110  0.6592  0.9590  0.00000
  4 000101100000000000101000011110  0.0859  0.0790  0.66797
  5 100010100110011001000000111100  0.5406  0.1268  0.00000

 Average Values:                     0.3954  0.3387  0.19268

 Average Function Value of Generation= 0.19268
 Maximum Function Value             = 0.66797


  Number of Crossovers       =   77
  Elitist Reproduction on Individual     3


################# Generation   43  #################
  #       Binary Code                Param1  Param2  Fitness
```

```
 1 000101100010000001101010001110  0.0864  0.2075  0.04268
 2 000101100111001010100100011110  0.0877  0.3212  0.02359
 3 000101100000000000101000011110  0.0859  0.0790  0.66797
 4 000101100100000011100000001110  0.0869  0.4379  0.28284
 5 100100100011001010101010011110  0.5711  0.3330  0.00000

Average Values:                    0.1836  0.2757  0.20342

Average Function Value of Generation= 0.20342
Maximum Function Value              = 0.66797


 Number of Crossovers       =   79
 Elitist Reproduction on Individual    1


################  Generation   44  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000101100000000000101000011110  0.0859  0.0790  0.66797
 2 000101100110000001101000001110  0.0874  0.2036  0.02538
 3 000101100110000010100000011110  0.0874  0.3134  0.05935
 4 000101100100000011101010001110  0.0869  0.4575  0.37484
 5 000101100000000011101000001110  0.0859  0.4536  0.38353

Average Values:                    0.0867  0.3014  0.30221

Average Function Value of Generation= 0.30221
Maximum Function Value              = 0.66797


 Number of Crossovers       =   82


################  Generation   45  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000101100000000000101000011110  0.0859  0.0790  0.66797
 2 000101100000000001101010001110  0.0859  0.2075  0.04333
 3 000101100000000000101000001110  0.0859  0.0786  0.67412
 4 000101100000000011101010001110  0.0859  0.4575  0.38651
 5 000101100000000011101000001110  0.0859  0.4536  0.38353

Average Values:                    0.0859  0.2552  0.43109

Average Function Value of Generation= 0.43109
Maximum Function Value              = 0.67412


 Number of Crossovers       =   68


%%%%%%  Restart micro-population at generation   45  %%%%%%


################  Generation   46  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000101100000000000101000001110  0.0859  0.0786  0.67412
 2 001000001110001010100011111011  0.1319  0.6405  0.00328
 3 111000001111001010000110100001  0.8824  0.5127  0.00082
 4 100111000111010111101100011000  0.6112  0.9617  0.00000
 5 001001001110001101101011101000  0.1441  0.7102  0.00001

Average Values:                    0.3711  0.5807  0.13565

Average Function Value of Generation= 0.13565
```

Maximum Function Value = 0.67412

Number of Crossovers = 81
Elitist Reproduction on Individual 1

################# Generation 47 #################

#	Binary Code	Param1	Param2	Fitness
1	000101100000000000101000001110	0.0859	0.0786	0.67412
2	000100010000000101000001001010	0.0664	0.6273	0.13848
3	110101111110000000000000001000	0.8433	0.0002	0.00087
4	001101001100000001000101111011	0.2061	0.1366	0.00034
5	111000011100010100000110101011	0.8819	0.5130	0.00082
Average Values:		0.4167	0.2711	0.16292

Average Function Value of Generation= 0.16292
Maximum Function Value = 0.67412

Number of Crossovers = 66
Elitist Reproduction on Individual 1

################# Generation 48 #################

#	Binary Code	Param1	Param2	Fitness
1	000101100000000000101000001110	0.0859	0.0786	0.67412
2	000100010000000101000000101011	0.0664	0.6263	0.13309
3	000100000000000000001000001010	0.0625	0.0159	0.10110
4	010101111000000000000001001010	0.3418	0.0023	0.00001
5	000101110000000101101000001010	0.0898	0.7035	0.01507
Average Values:		0.1293	0.2853	0.18468

Average Function Value of Generation= 0.18468
Maximum Function Value = 0.67412

Number of Crossovers = 70
Elitist Reproduction on Individual 4

################# Generation 49 #################

#	Binary Code	Param1	Param2	Fitness
1	000100110000000101100000101010	0.0742	0.6888	0.07124
2	000100000000000100001000001010	0.0625	0.5159	0.02154
3	000100100000000000001000001110	0.0703	0.0161	0.10290
4	000101100000000000101000001110	0.0859	0.0786	0.67412
5	000101010000000101101000101110	0.0820	0.7046	0.01712
Average Values:		0.0750	0.4008	0.17738

Average Function Value of Generation= 0.17738
Maximum Function Value = 0.67412

Number of Crossovers = 68
Elitist Reproduction on Individual 3

################# Generation 50 #################

#	Binary Code	Param1	Param2	Fitness
1	000101110000000000100000101010	0.0898	0.0638	0.65246

```
  2 00010010000000000100000101010  0.0703  0.0638  0.98354
  3 00010110000000000101000001110  0.0859  0.0786  0.67412
  4 00010011000000000100000101110  0.0742  0.0639  0.95216
  5 00010110000000000001000001110  0.0859  0.0161  0.07793

 Average Values:                   0.0813  0.0572  0.66804

 Average Function Value of Generation= 0.66804
 Maximum Function Value              = 0.98354


  Number of Crossovers       =   66
  Elitist Reproduction on Individual    1


#################  Generation   51  #################
  #       Binary Code              Param1  Param2  Fitness
  1 00010010000000000100000101010  0.0703  0.0638  0.98354
  2 00010110000000000100000101110  0.0859  0.0639  0.74534
  3 00010111000000000100000101010  0.0898  0.0638  0.65246
  4 00010110000000000100000101010  0.0859  0.0638  0.74492
  5 00010110000000000100000001010  0.0859  0.0628  0.74094

 Average Values:                   0.0836  0.0636  0.77344

 Average Function Value of Generation= 0.77344
 Maximum Function Value              = 0.98354


  Number of Crossovers       =   70
  Elitist Reproduction on Individual    1

%%%%%%%  Restart micro-population at generation   51  %%%%%%%

#################  Generation   52  #################
  #       Binary Code              Param1  Param2  Fitness
  1 00010010000000000100000101010  0.0703  0.0638  0.98354
  2 11001011000100001011101110000  0.7932  0.3662  0.00000
  3 10111000000001101111010001100l  0.7189  0.4773  0.00118
  4 10001010101101101100010000110l  0.5419  0.3832  0.00000
  5 01100100110011100110100101101O  0.3938  0.2059  0.00048

 Average Values:                   0.5036  0.2993  0.19704

 Average Function Value of Generation= 0.19704
 Maximum Function Value              = 0.98354


  Number of Crossovers       =   78
  Elitist Reproduction on Individual    2


#################  Generation   53  #################
  #       Binary Code              Param1  Param2  Fitness
  1 00111010000001001111010000101l  0.2266  0.4769  0.11450
  2 00010010000000000100000101010  0.0703  0.0638  0.98354
  3 01100010000011000010000001010  0.3830  0.0628  0.00110
  4 00110000000000001100100111000  0.1875  0.1970  0.00002
  5 00000010100001101000000001000  0.0099  0.2503  0.03934

 Average Values:                   0.1755  0.2102  0.22770
```

Average Function Value of Generation= 0.22770
Maximum Function Value = 0.98354

Number of Crossovers = 83
Elitist Reproduction on Individual 2

################ Generation 54 ################

#	Binary Code	Param1	Param2	Fitness
1	0010101010000100111101000010000	0.1661	0.4768	0.00000
2	0001001000000000000100000101010	0.0703	0.0638	0.98354
3	0011101000000000001100000101010	0.2266	0.1888	0.00078
4	0000001000000010000000000101000	0.0079	0.0012	0.00059
5	0001101000000000010110100101011	0.1016	0.3529	0.00000
Average Values:		0.1145	0.2167	0.19698

Average Function Value of Generation= 0.19698
Maximum Function Value = 0.98354

Number of Crossovers = 78
Elitist Reproduction on Individual 3

################ Generation 55 ################

#	Binary Code	Param1	Param2	Fitness
1	0001101000000000011110000101010	0.1016	0.4700	0.16761
2	0001101000000000010100100101010	0.1016	0.3216	0.01180
3	0001001000000000000100000101010	0.0703	0.0638	0.98354
4	0010001000000010000000000101010	0.1329	0.0013	0.00020
5	0001001000000000000110100101011	0.0703	0.1029	0.34068
Average Values:		0.0953	0.1919	0.30077

Average Function Value of Generation= 0.30077
Maximum Function Value = 0.98354

Number of Crossovers = 80
Elitist Reproduction on Individual 4

################ Generation 56 ################

#	Binary Code	Param1	Param2	Fitness
1	0001101000000000011100000101010	0.1016	0.4388	0.14865
2	0001101000000000000100000101010	0.1016	0.0638	0.37014
3	0001101000000000001110000101010	0.1016	0.2200	0.07195
4	0001001000000000000100000101010	0.0703	0.0638	0.98354
5	0001101000000000001100000101010	0.1016	0.1888	0.00095
Average Values:		0.0953	0.1950	0.31505

Average Function Value of Generation= 0.31505
Maximum Function Value = 0.98354

Number of Crossovers = 70

################ Generation 57 ################

#	Binary Code	Param1	Param2	Fitness
1	0001101000000000000100000101010	0.1016	0.0638	0.37014

```
 2 0001101000000000111000001010l0  0.1016  0.4388  0.14865
 3 000100100000000001100000101010  0.0703  0.1888  0.00254
 4 000110100000000000100000101010  0.1016  0.0638  0.37014
 5 000100100000000000100000101010  0.0703  0.0638  0.98354

Average Values:                    0.0891  0.1638  0.37500

Average Function Value of Generation= 0.37500
Maximum Function Value              = 0.98354


 Number of Crossovers      =   75


%%%%%%%  Restart micro-population at generation   57  %%%%%%%


#################  Generation   58  #################
 #      Binary Code                Param1  Param2  Fitness
 1 000100100000000000100000101010  0.0703  0.0638  0.98354
 2 111111111000101011110110110010  0.9982  0.4820  0.00091
 3 111110000101100111110101101110  0.9701  0.9799  0.00000
 4 011000000110101010111101101110  0.3766  0.3706  0.00000
 5 000011111100101110100011001000  0.0617  0.8186  0.03601

Average Values:                    0.4954  0.5430  0.20409

Average Function Value of Generation= 0.20409
Maximum Function Value              = 0.98354


 Number of Crossovers      =   62
 Elitist Reproduction on Individual    1


#################  Generation   59  #################
 #      Binary Code                Param1  Param2  Fitness
 1 000100100000000000100000101010  0.0703  0.0638  0.98354
 2 000011101100101000100001101000  0.0578  0.0657  0.93723
 3 000010100100100100100011101010  0.0402  0.5697  0.00001
 4 000110100101100101110101101110  0.1029  0.7299  0.00012
 5 111110100000100010100100100010  0.9767  0.3214  0.00001

Average Values:                    0.2496  0.3501  0.38418

Average Function Value of Generation= 0.38418
Maximum Function Value              = 0.98354


 Number of Crossovers      =   71
 Elitist Reproduction on Individual    4


#################  Generation   60  #################
 #      Binary Code                Param1  Param2  Fitness
 1 000010100100100100100011101010  0.0402  0.5697  0.00001
 2 000110100100100100100010101010  0.1027  0.5677  0.00000
 3 000100100000100001100000101010  0.0704  0.1888  0.00253
 4 000100100000000000100000101010  0.0703  0.0638  0.98354
 5 000010101100100000100011101000  0.0421  0.0696  0.61152

Average Values:                    0.0651  0.2919  0.31952

Average Function Value of Generation= 0.31952
```

Maximum Function Value = 0.98354

Number of Crossovers = 79
Elitist Reproduction on Individual 2

################# Generation 61 #################

#	Binary Code	Param1	Param2	Fitness
1	000100101100100001100010101010	0.0734	0.1927	0.00580
2	000100100000000000100000101010	0.0703	0.0638	0.98354
3	000010101100100000100011101010	0.0421	0.0696	0.61136
4	000100100100000000100010101010	0.0713	0.0677	0.98414
5	000100100000000001100000101010	0.0703	0.1888	0.00254
Average Values:		0.0655	0.1165	0.51748

Average Function Value of Generation= 0.51748
Maximum Function Value = 0.98414

Number of Crossovers = 71

################# Generation 62 #################

#	Binary Code	Param1	Param2	Fitness
1	000100100100100001100010101010	0.0714	0.1927	0.00590
2	000100100100000000100000101010	0.0713	0.0638	0.97764
3	000100100000000000100010101010	0.0703	0.0677	0.99008
4	000100100100000000100010101010	0.0713	0.0677	0.98414
5	000000101100100000100010101010	0.0109	0.0677	0.05831
Average Values:		0.0590	0.0919	0.60321

Average Function Value of Generation= 0.60321
Maximum Function Value = 0.99008

Number of Crossovers = 85

%%%%%%% Restart micro-population at generation 62 %%%%%%%

################# Generation 63 #################

#	Binary Code	Param1	Param2	Fitness
1	000100100000000000100010101010	0.0703	0.0677	0.99008
2	111011111101110111100000000100	0.9370	0.9377	0.00000
3	111011010011100010010011110101	0.9267	0.2887	0.00004
4	110110011100011100011100111101	0.8507	0.5566	0.00000
5	110101010001111100001001111010	0.8325	0.5194	0.00083
Average Values:		0.7234	0.4740	0.19819

Average Function Value of Generation= 0.19819
Maximum Function Value = 0.99008

Number of Crossovers = 77
Elitist Reproduction on Individual 5

################# Generation 64 #################

#	Binary Code	Param1	Param2	Fitness

```
  1 1001000100001011001010111111010  0.5666  0.5858  0.00000
  2 0111000000100000100100101001O1  0.4380  0.2863  0.20530
  3 0001000100010010000010101110I0  0.0667  0.0213  0.17015
  4 1101001100001000000000000111010  0.8244  0.0018  0.00076
  5 0001001000000000001000101010I0  0.0703  0.0677  0.99008

 Average Values:                     0.3932  0.1926  0.27326

 Average Function Value of Generation= 0.27326
 Maximum Function Value              = 0.99008


  Number of Crossovers       =   77
  Elitist Reproduction on Individual    4


################# Generation   65  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0001000100000000001010101010I0  0.0664  0.0833  0.80811
  2 0101001000001000001000101110I0  0.3204  0.0682  0.03682
  3 0111000100010010100110101010I0  0.4417  0.3021  0.09533
  4 0001001000000000001000101010I0  0.0703  0.0677  0.99008
  5 1111000100000000100100101001O1  0.9414  0.2863  0.00000

 Average Values:                     0.3681  0.1615  0.38607

 Average Function Value of Generation= 0.38607
 Maximum Function Value              = 0.99008


  Number of Crossovers       =   77
  Elitist Reproduction on Individual    3


################# Generation   66  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0011001000000000000000101010I0  0.1953  0.0052  0.00027
  2 0101001000001000001000101110I0  0.3204  0.0682  0.03682
  3 0001001000000000001000101010I0  0.0703  0.0677  0.99008
  4 0001000100000000001000101010I0  0.0664  0.0677  0.99929
  5 0001000100000000001000101010I0  0.0664  0.0677  0.99929

 Average Values:                     0.1438  0.0553  0.60515

 Average Function Value of Generation= 0.60515
 Maximum Function Value              = 0.99929


  Number of Crossovers       =   64


%%%%%%%  Restart micro-population at generation   66  %%%%%%%


################# Generation   67  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0001000100000000001000101010I0  0.0664  0.0677  0.99929
  2 1100100001000000000100101000O0  0.7823  0.0181  0.00011
  3 0011110000000010111011111010O1  0.2344  0.4681  0.21614
  4 0010011011001111010100001001O1  0.1516  0.6574  0.00002
  5 1010101010101100111111100011O0  0.6667  0.4965  0.02681

 Average Values:                     0.3803  0.3415  0.24847
```

```
 Average Function Value of Generation= 0.24847
 Maximum Function Value              = 0.99929


  Number of Crossovers       =   59
  Elitist Reproduction on Individual    4


#################  Generation   68  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 100011000100001010001001001001  0.5479  0.2679  0.00000
  2 000100000000001011100011101001  0.0625  0.4446  0.45230
  3 001111000010111011101110001001  0.2351  0.4651  0.23296
  4 000100010000000000100010101010  0.0664  0.0677  0.99929
  5 101010100000010011110010101000  0.6641  0.4739  0.08201

 Average Values:                      0.3152  0.3438  0.35331

 Average Function Value of Generation= 0.35331
 Maximum Function Value              = 0.99929


  Number of Crossovers       =   80
  Elitist Reproduction on Individual    4


#################  Generation   69  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 001100000010111011101010001001  0.1882  0.4573  0.00115
  2 100110110000000001110010101000  0.6055  0.2239  0.00859
  3 000000010000010000100010101010  0.0040  0.0677  0.02283
  4 000100010000000000100010101010  0.0664  0.0677  0.99929
  5 001110000000001001100110101001  0.2188  0.2005  0.00401

 Average Values:                      0.2166  0.2034  0.20718

 Average Function Value of Generation= 0.20718
 Maximum Function Value              = 0.99929


  Number of Crossovers       =   67


#################  Generation   70  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 100100010000010001110010101000  0.5665  0.2239  0.00000
  2 000100010000000000100010101010  0.0664  0.0677  0.99929
  3 000100110000000001110010101000  0.0742  0.2239  0.24466
  4 000110110000000000110010101000  0.1055  0.0989  0.12604
  5 001110000000000001100110101011  0.2188  0.2005  0.00404

 Average Values:                      0.2063  0.1630  0.27481

 Average Function Value of Generation= 0.27481
 Maximum Function Value              = 0.99929


  Number of Crossovers       =   76
  Elitist Reproduction on Individual    1


#################  Generation   71  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010000000000100010101010  0.0664  0.0677  0.99929
```

```
 2 0001000100000000011001101010 11  0.0664  0.2005  0.02316
```

```
 2 000100010000000001100110101011  0.0664  0.2005  0.02316
 3 000100000000000001000101010 11  0.0625  0.0677  0.98497
 4 001100000000000001100110101001  0.1875  0.2005  0.00004
 5 000100010000000001110010101010  0.0664  0.2239  0.25628

Average Values:                    0.0898  0.1521  0.45275

Average Function Value of Generation= 0.45275
Maximum Function Value              = 0.99929


 Number of Crossovers      =   71


################# Generation   72  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010000000001001101010 11  0.0664  0.0755  0.94273
 2 000100010000000001100110101011  0.0664  0.2005  0.02316
 3 000100010000000001000101010 10  0.0664  0.0677  0.99929
 4 000100000000000001000101010 10  0.0625  0.0677  0.98501
 5 000100010000000001100101010 10  0.0664  0.0989  0.43330

Average Values:                    0.0656  0.1021  0.67670

Average Function Value of Generation= 0.67670
Maximum Function Value              = 0.99929


 Number of Crossovers      =   75
 Elitist Reproduction on Individual    1


################# Generation   73  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010000000001000101010 10  0.0664  0.0677  0.99929
 2 000100010000000001001101010 11  0.0664  0.0755  0.94273
 3 000100010000000001100101010 10  0.0664  0.0989  0.43330
 4 000100010000000001100101010 10  0.0664  0.0989  0.43330
 5 000100010000000001001101010 11  0.0664  0.0755  0.94273

Average Values:                    0.0664  0.0833  0.75027

Average Function Value of Generation= 0.75027
Maximum Function Value              = 0.99929


 Number of Crossovers      =   80
 Elitist Reproduction on Individual    5


################# Generation   74  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010000000001001101010 11  0.0664  0.0755  0.94273
 2 000100010000000001000101010 11  0.0664  0.0677  0.99925
 3 000100010000000001001101010 11  0.0664  0.0755  0.94273
 4 000100010000000001001101010 11  0.0664  0.0755  0.94273
 5 000100010000000001000101010 10  0.0664  0.0677  0.99929

Average Values:                    0.0664  0.0724  0.96535

Average Function Value of Generation= 0.96535
Maximum Function Value              = 0.99929
```

```
 Number of Crossovers      =   68
 Elitist Reproduction on Individual    4


################# Generation   75  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010000000000100110101011  0.0664  0.0755  0.94273
 2 000100010000000000100110101011  0.0664  0.0755  0.94273
 3 000100010000000000100110101011  0.0664  0.0755  0.94273
 4 000100010000000000100010101010  0.0664  0.0677  0.99929
 5 000100010000000000100010101011  0.0664  0.0677  0.99925

Average Values:                    0.0664  0.0724  0.96535

Average Function Value of Generation= 0.96535
Maximum Function Value              = 0.99929


 Number of Crossovers      =   72
 Elitist Reproduction on Individual    3


%%%%%%%  Restart micro-population at generation   75  %%%%%%%


################# Generation   76  #################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010000000000100010101010  0.0664  0.0677  0.99929
 2 101111110101111001000001000010  0.7476  0.1270  0.00000
 3 011001110111111000011111001101  0.4043  0.0609  0.04075
 4 110100001100110001001000010010  0.8156  0.1412  0.00008
 5 110100001010100111011000011101  0.8151  0.9228  0.00001

Average Values:                    0.5698  0.2639  0.20803

Average Function Value of Generation= 0.20803
Maximum Function Value              = 0.99929


 Number of Crossovers      =   72
 Elitist Reproduction on Individual    3


################# Generation   77  #################
 #       Binary Code                Param1  Param2  Fitness
 1 010100010000110001001010111010  0.3166  0.1463  0.00004
 2 010000101010110011011111001101  0.2604  0.4360  0.30846
 3 000100010000000000100010101010  0.0664  0.0677  0.99929
 4 010000110111101000001111001101  0.2636  0.0297  0.27098
 5 110100001000110000000010100010  0.8147  0.0049  0.00079

Average Values:                    0.3443  0.1369  0.31591

Average Function Value of Generation= 0.31591
Maximum Function Value              = 0.99929


 Number of Crossovers      =   83
 Elitist Reproduction on Individual    2


################# Generation   78  #################
 #       Binary Code                Param1  Param2  Fitness
 1 010100101010110010101110101010  0.3229  0.3411  0.00002
```

```
  2 0001000100000000000100010101010  0.0664  0.0677  0.99929
  3 0001000000000000000010110101001  0.0625  0.0442  0.65745
  4 0001000101111000001010111101111  0.0682  0.0854  0.76094
  5 1101000010000100001000101010 10  0.8145  0.0677  0.02958

 Average Values:                     0.2669  0.1212  0.48946

 Average Function Value of Generation= 0.48946
 Maximum Function Value              = 0.99929


  Number of Crossovers        =   75
  Elitist Reproduction on Individual    4


#################  Generation   79  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0001000100000000000100110101010  0.0664  0.0755  0.94312
  2 0001000100011000001010111101010  0.0668  0.0853  0.76555
  3 0001000100111000001010101011110  0.0673  0.0834  0.80553
  4 0001000100000000000100010101010  0.0664  0.0677  0.99929
  5 0001000100110000000100010101010  0.0671  0.0677  0.99936

 Average Values:                     0.0668  0.0759  0.90257

 Average Function Value of Generation= 0.90257
 Maximum Function Value              = 0.99936


  Number of Crossovers        =   64
  Elitist Reproduction on Individual    4


#################  Generation   80  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0001000100010000000100010101010  0.0667  0.0677  0.99940
  2 0001000100000000000100010101010  0.0664  0.0677  0.99929
  3 0001000100001000001010101011110  0.0665  0.0834  0.80558
  4 0001000100110000000100010101010  0.0671  0.0677  0.99936
  5 0001000100010000000100010101010  0.0667  0.0677  0.99940

 Average Values:                     0.0667  0.0708  0.96061

 Average Function Value of Generation= 0.96061
 Maximum Function Value              = 0.99940


  Number of Crossovers        =   72


%%%%%%%  Restart micro-population at generation   80  %%%%%%%


#################  Generation   81  #################
  #       Binary Code                Param1  Param2  Fitness
  1 0001000100010000000100010101010  0.0667  0.0677  0.99940
  2 1101110110011110011000100111 01  0.8657  0.1923  0.00030
  3 0010000111011100101000101111100  0.1323  0.3182  0.00073
  4 1111110001100000010100110000 11  0.9859  0.1622  0.00000
  5 1000001010111000101001001100 11  0.5106  0.3219  0.00125

 Average Values:                     0.5122  0.2125  0.20034

 Average Function Value of Generation= 0.20034
```

Maximum Function Value = 0.99940

Number of Crossovers = 73
Elitist Reproduction on Individual 3

################# Generation 82 #################

#	Binary Code	Param1	Param2	Fitness
1	100000011001000010100100101011	0.5061	0.3216	0.00204
2	101000111111100010100000110110	0.6405	0.3142	0.01565
3	000100010001000000100010101010	0.0667	0.0677	0.99940
4	000100111001000010100010101010	0.0764	0.3177	0.04817
5	110111111001110001100010111001	0.8735	0.1932	0.00026
Average Values:		0.4326	0.2429	0.21310

Average Function Value of Generation= 0.21310
Maximum Function Value = 0.99940

Number of Crossovers = 75
Elitist Reproduction on Individual 2

################# Generation 83 #################

#	Binary Code	Param1	Param2	Fitness
1	001100111101000010100010101010	0.2024	0.3177	0.00157
2	000100010001000000100010101010	0.0667	0.0677	0.99940
3	100100011011100010100010101010	0.5692	0.3177	0.00000
4	100000011111100000100010111110	0.5077	0.0683	0.05532
5	100000011001100000100000110110	0.5062	0.0641	0.06350
Average Values:		0.3704	0.1671	0.22396

Average Function Value of Generation= 0.22396
Maximum Function Value = 0.99940

Number of Crossovers = 74
Elitist Reproduction on Individual 1

################# Generation 84 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001000000100010101010	0.0667	0.0677	0.99940
2	000100011001000000100000100110	0.0686	0.0637	0.98983
3	100000010011000000100010101010	0.5047	0.0677	0.07389
4	100000011011100000100010110110	0.5067	0.0681	0.06088
5	100000010111100000100010111010	0.5058	0.0682	0.06676
Average Values:		0.3305	0.0671	0.43815

Average Function Value of Generation= 0.43815
Maximum Function Value = 0.99940

Number of Crossovers = 81

################# Generation 85 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001000000100010101110	0.0667	0.0678	0.99923
2	000100010011000000100010101010	0.0671	0.0677	0.99936

```
  3 100100011011000000100000100110  0.5691  0.0637  0.00002
  4 000100010001000000100010101010  0.0667  0.0677  0.99940
  5 000100010001000000100010101010  0.0667  0.0677  0.99940

 Average Values:                    0.1672  0.0669  0.79948

 Average Function Value of Generation= 0.79948
 Maximum Function Value              = 0.99940


  Number of Crossovers      =   75


%%%%%%%  Restart micro-population at generation   85  %%%%%%%


#################  Generation   86  #################
  #      Binary Code                Param1  Param2  Fitness
  1 000100010001000000100010101010  0.0667  0.0677  0.99940
  2 111101010110111111101011100011  0.9587  0.9601  0.00000
  3 100011110110000111100000010001  0.5601  0.9380  0.00000
  4 110111111101100000000000101101  0.8744  0.0014  0.00058
  5 100011010001110001001010000010  0.5512  0.1446  0.00000

 Average Values:                    0.6022  0.4224  0.20000

 Average Function Value of Generation= 0.20000
 Maximum Function Value              = 0.99940


  Number of Crossovers      =   81
  Elitist Reproduction on Individual    5

#################  Generation   87  #################
  #      Binary Code                Param1  Param2  Fitness
  1 110011011101100001000000001011  0.8041  0.1253  0.00060
  2 100111010101000000000000101110  0.6145  0.0014  0.00105
  3 100100010001100001101010101010  0.5668  0.2083  0.00000
  4 110111110110111010101011101001  0.8728  0.3352  0.00012
  5 000100010001000000100010101010  0.0667  0.0677  0.99940

 Average Values:                    0.5850  0.1476  0.20024

 Average Function Value of Generation= 0.20024
 Maximum Function Value              = 0.99940


  Number of Crossovers      =   87
  Elitist Reproduction on Individual    5


#################  Generation   88  #################
  #      Binary Code                Param1  Param2  Fitness
  1 100101010001000000100000101010  0.5823  0.0638  0.00117
  2 100111011101000001000000001111  0.6165  0.1255  0.00329
  3 000110010101000000000010101010  0.0989  0.0052  0.01192
  4 010100010101000000000010001011  0.3176  0.0042  0.00124
  5 000100010001000000100010101010  0.0667  0.0677  0.99940

 Average Values:                    0.3364  0.0533  0.20341

 Average Function Value of Generation= 0.20341
 Maximum Function Value              = 0.99940
```

```
 Number of Crossovers       =   81
 Elitist Reproduction on Individual    5


################# Generation   89  #################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010101000001000010101110  0.0676  0.1303  0.02076
  2 000100010101000000000010101010  0.0676  0.0052  0.02741
  3 100101010001000001100000001110  0.5823  0.1879  0.00000
  4 000100010001000000000010101010  0.0667  0.0052  0.02742
  5 000100010001000001000010101010  0.0667  0.0677  0.99940

 Average Values:                      0.1702  0.0793  0.21500

 Average Function Value of Generation= 0.21500
 Maximum Function Value              = 0.99940


 Number of Crossovers       =   75


%%%%%%%  Restart micro-population at generation   89  %%%%%%%


################# Generation   90  #################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001000001000010101010  0.0667  0.0677  0.99940
  2 111011100110001000000111101010  0.9312  0.0150  0.00000
  3 000110001000010101011000100101  0.0958  0.6730  0.08067
  4 001110000010101100101100011001  0.2194  0.5867  0.00052
  5 000001010000000000101010000011  0.0195  0.0821  0.12108

 Average Values:                      0.2665  0.2849  0.24034

 Average Function Value of Generation= 0.24034
 Maximum Function Value              = 0.99940


 Number of Crossovers       =   65
 Elitist Reproduction on Individual    1


################# Generation   91  #################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001000001000010101010  0.0667  0.0677  0.99940
  2 000001010001000001000010000011  0.0198  0.0665  0.14865
  3 000100010001000100100100001001  0.0667  0.5706  0.00003
  4 001110010001001000101000111001  0.2229  0.0799  0.20940
  5 000000010001000000101010001010  0.0042  0.0823  0.01946

 Average Values:                      0.0760  0.1734  0.27539

 Average Function Value of Generation= 0.27539
 Maximum Function Value              = 0.99940


 Number of Crossovers       =   72
 Elitist Reproduction on Individual    4


################# Generation   92  #################
  #      Binary Code                  Param1  Param2  Fitness
```

```
 1 001110010001000000100010111010  0.2229  0.0682  0.23824
 2 001100010001001000100000101001  0.1917  0.0638  0.00485
 3 001011010001001000100000101011  0.1761  0.0638  0.00003
 4 000100010001000000100010101010  0.0667  0.0677  0.99940
 5 000110010001001000101000111001  0.0979  0.0799  0.40061

Average Values:                    0.1510  0.0687  0.32863

Average Function Value of Generation= 0.32863
Maximum Function Value              = 0.99940


 Number of Crossovers       =   76
 Elitist Reproduction on Individual    5


################# Generation   93  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010001001000101010111010  0.0667  0.0838  0.79780
 2 001110010001000000100000111001  0.2229  0.0642  0.23733
 3 000100010001000000100010101000  0.0667  0.0676  0.99948
 4 001110010001000000101000111001  0.2229  0.0799  0.20895
 5 000100010001000000100010101010  0.0667  0.0677  0.99940

Average Values:                    0.1292  0.0726  0.64860

Average Function Value of Generation= 0.64860
Maximum Function Value              = 0.99948


 Number of Crossovers       =   78
 Elitist Reproduction on Individual    3


################# Generation   94  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010001000000101010101010  0.0667  0.0833  0.80821
 2 001100010001000000100000101000  0.1917  0.0637  0.00481
 3 000100010001000000100010101000  0.0667  0.0676  0.99948
 4 001110010001000000100000101010  0.2229  0.0638  0.23686
 5 000100010001000000101010111000  0.0667  0.0837  0.79911

Average Values:                    0.1229  0.0724  0.56969

Average Function Value of Generation= 0.56969
Maximum Function Value              = 0.99948


 Number of Crossovers       =   75


%%%%%%%  Restart micro-population at generation   94  %%%%%%%


################# Generation   95  #################
 #       Binary Code               Param1  Param2  Fitness
 1 000100010001000000100010101000  0.0667  0.0676  0.99948
 2 001111011011111110101101110010  0.2412  0.6832  0.06203
 3 010111011111111110111011011010  0.3672  0.8661  0.00000
 4 011110110010111000101011101101  0.4812  0.0854  0.24615
 5 100101000001111110010010000010  0.5786  0.7852  0.00000

Average Values:                    0.3470  0.4975  0.26153
```

Average Function Value of Generation= 0.26153
Maximum Function Value = 0.99948

Number of Crossovers = 87
Elitist Reproduction on Individual 4

################# Generation 96 #################

#	Binary Code	Param1	Param2	Fitness
1	000111110010111010111010100100	0.1218	0.3644	0.00000
2	100100000001011110110010101010	0.5629	0.8490	0.00000
3	000110010011001000100110100000	0.0984	0.0752	0.42211
4	000100010001000000100010101000	0.0667	0.0676	0.99948
5	000111010011000000111101110010	0.1140	0.1207	0.01099
Average Values:		0.1927	0.2954	0.28652

Average Function Value of Generation= 0.28652
Maximum Function Value = 0.99948

Number of Crossovers = 77
Elitist Reproduction on Individual 5

################# Generation 97 #################

#	Binary Code	Param1	Param2	Fitness
1	000110010001001000100010101000	0.0979	0.0676	0.45717
2	000110010011001000111101110000	0.0984	0.1206	0.03357
3	000100010011001000100010100000	0.0672	0.0674	0.99967
4	100100010001001110110010101010	0.5667	0.8490	0.00000
5	000100010001000000100010101000	0.0667	0.0676	0.99948
Average Values:		0.1794	0.2344	0.49798

Average Function Value of Generation= 0.49798
Maximum Function Value = 0.99967

Number of Crossovers = 67
Elitist Reproduction on Individual 1

################# Generation 98 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010011001000100010100000	0.0672	0.0674	0.99967
2	000100010011001000100010101000	0.0672	0.0676	0.99942
3	000100010011001000100010101000	0.0672	0.0676	0.99942
4	000110010001000000111100111000	0.0979	0.1189	0.04160
5	000100010011001000100010101000	0.0672	0.0676	0.99942
Average Values:		0.0733	0.0778	0.80791

Average Function Value of Generation= 0.80791
Maximum Function Value = 0.99967

Number of Crossovers = 72

%%%%%%% Restart micro-population at generation 98 %%%%%%%

```
#################  Generation   99  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010011001000100010100000  0.0672  0.0674  0.99967
  2 110000000000011100101001101100  0.7501  0.5814  0.00000
  3 111100010011111100110011011001  0.9424  0.6004  0.00000
  4 010110000000110110011000010 01  0.3439  0.3987  0.00001
  5 100101101001110100000110000011  0.5883  0.5118  0.00014

 Average Values:                     0.5384  0.4320  0.19996

 Average Function Value of Generation= 0.19996
 Maximum Function Value             = 0.99967


  Number of Crossovers       =   81
  Elitist Reproduction on Individual    1


#################  Generation  100  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010011001000100010100000  0.0672  0.0674  0.99967
  2 010110000011001001100100101000  0.3445  0.1965  0.00000
  3 000100111001101100100110000000  0.0766  0.5742  0.00012
  4 010110000011011011101000101001  0.3446  0.4544  0.00011
  5 010110010011011000101010100001  0.3485  0.0830  0.00004

 Average Values:                     0.2363  0.2751  0.19999

 Average Function Value of Generation= 0.19999
 Maximum Function Value             = 0.99967


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    5


#################  Generation  101  #################
  #       Binary Code                Param1  Param2  Fitness
  1 010100010011001011101010101001  0.3172  0.4583  0.02828
  2 010100010011011001101010101000  0.3172  0.2083  0.00346
  3 010100000011011000100010101000  0.3133  0.0676  0.08362
  4 000110111011011100101110100001  0.1082  0.5909  0.00136
  5 000100010011001000100010100000  0.0672  0.0674  0.99967

 Average Values:                     0.2246  0.2785  0.22328

 Average Function Value of Generation= 0.22328
 Maximum Function Value             = 0.99967


  Number of Crossovers       =   68
  Elitist Reproduction on Individual    4


#################  Generation  102  #################
  #       Binary Code                Param1  Param2  Fitness
  1 010100010011001001100010101000  0.3172  0.1926  0.00033
  2 000100000011001000100010101000  0.0633  0.0676  0.98970
  3 010100000011011001100010101000  0.3133  0.1926  0.00050
  4 000100010011001000100010100000  0.0672  0.0674  0.99967
  5 000100000011011000100010100000  0.0633  0.0674  0.99029

 Average Values:                     0.1649  0.1175  0.59610
```

```
Average Function Value of Generation= 0.59610
Maximum Function Value              = 0.99967


 Number of Crossovers       =   80
 Elitist Reproduction on Individual    5


################  Generation  103  ################
  #       Binary Code                  Param1  Param2  Fitness
  1 00010001001101100010001010000O     0.0672  0.0674  0.99964
  2 01010000001101100110001010100O     0.3133  0.1926  0.00050
  3 00010001001101100010001010000O     0.0672  0.0674  0.99964
  4 00010000001100100010001010000O     0.0633  0.0674  0.98996
  5 00010001001100100010001010000O     0.0672  0.0674  0.99967

 Average Values:                       0.1156  0.0924  0.79788

 Average Function Value of Generation= 0.79788
 Maximum Function Value              = 0.99967


  Number of Crossovers       =   74

%%%%%%%  Restart micro-population at generation  103  %%%%%%%

################  Generation  104  ################
  #       Binary Code                  Param1  Param2  Fitness
  1 000100010011001000100010100000     0.0672  0.0674  0.99967
  2 001110000000000110110101011010     0.2188  0.8543  0.01177
  3 101110001001101110111000001101     0.7211  0.8598  0.00013
  4 010111001001101010000101010110     0.3617  0.2604  0.00000
  5 101111101011101010110101010100     0.7450  0.3541  0.00000

 Average Values:                       0.4228  0.4792  0.20232

 Average Function Value of Generation= 0.20232
 Maximum Function Value              = 0.99967


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    4


################  Generation  105  ################
  #       Binary Code                  Param1  Param2  Fitness
  1 010111000001110100000001001001 10  0.3446  0.0090  0.00001
  2 100100011001001010111000000100     0.5687  0.3595  0.00000
  3 001110001001001110110000101000     0.2210  0.8450  0.01461
  4 000100010011001000100010100000     0.0672  0.0674  0.99967
  5 001110010010001100100111001010     0.2232  0.5765  0.00007

 Average Values:                       0.2849  0.3715  0.20287

 Average Function Value of Generation= 0.20287
 Maximum Function Value              = 0.99967


  Number of Crossovers       =   63
  Elitist Reproduction on Individual    1
```

################# Generation 106 #################

#	Binary Code	Param1	Param2	Fitness
1	0001000100110010001000101000000	0.0672	0.0674	0.99967
2	0001000100110010001001001000000	0.0672	0.0713	0.98461
3	0011000010110011101000001010000	0.1902	0.8137	0.00010
4	0001100000110011101100001010000	0.0945	0.8450	0.03811
5	0011000100100010001001111010100	0.1919	0.0775	0.00470
Average Values:		0.1222	0.3750	0.40544

Average Function Value of Generation= 0.40544
Maximum Function Value = 0.99967

Number of Crossovers = 73
Elitist Reproduction on Individual 4

################# Generation 107 #################

#	Binary Code	Param1	Param2	Fitness
1	0001000100110010001000001000000	0.0672	0.0635	0.99124
2	0001100000110010101001001000000	0.0945	0.3213	0.01785
3	0001000000110011001000101000000	0.0633	0.5674	0.00001
4	0001000100110010001000101000000	0.0672	0.0674	0.99967
5	0001100100110011001000001010000	0.0984	0.5637	0.00000
Average Values:		0.0781	0.3167	0.40175

Average Function Value of Generation= 0.40175
Maximum Function Value = 0.99967

Number of Crossovers = 71
Elitist Reproduction on Individual 1

%%%%%%% Restart micro-population at generation 107 %%%%%%%

################# Generation 108 #################

#	Binary Code	Param1	Param2	Fitness
1	0001000100110010001000101000000	0.0672	0.0674	0.99967
2	1010101001010111000111011100010	0.6654	0.5582	0.00000
3	0000000011011100010110000101100	0.0034	0.1726	0.00000
4	1010101100101001101011001111010	0.6686	0.8378	0.01197
5	1011010000110001101000011100010	0.7039	0.8160	0.00070
Average Values:		0.4217	0.4904	0.20247

Average Function Value of Generation= 0.20247
Maximum Function Value = 0.99967

Number of Crossovers = 81
Elitist Reproduction on Individual 1

################# Generation 109 #################

#	Binary Code	Param1	Param2	Fitness
1	0001000100110010001000101000000	0.0672	0.0674	0.99967
2	1001000100110011101000111000000	0.5672	0.8194	0.00000
3	0011000000110011101000001000000	0.1883	0.8135	0.00006
4	0001000001010110010010000100000	0.0638	0.1411	0.00255
5	0011000100110011101000111000000	0.1922	0.8194	0.00021

```
Average Values:                        0.2157  0.5321  0.20050

Average Function Value of Generation= 0.20050
Maximum Function Value               = 0.99967


 Number of Crossovers        =   78
 Elitist Reproduction on Individual    1


################# Generation  110  #################
 #      Binary Code                    Param1  Param2  Fitness
 1 0001000100110010001000101000000  0.0672  0.0674  0.99967
 2 0001000101010010010000000000000  0.0677  0.1250  0.04423
 3 0001000100110010101000111000000  0.0672  0.3193  0.04237
 4 0001000000110111000001000110000  0.0633  0.0171  0.11419
 5 0001000001010110001000000110000  0.0638  0.0640  0.98668

Average Values:                        0.0658  0.1186  0.43743

Average Function Value of Generation= 0.43743
Maximum Function Value               = 0.99967


 Number of Crossovers        =   74
 Elitist Reproduction on Individual    3


################# Generation  111  #################
 #      Binary Code                    Param1  Param2  Fitness
 1 0001000100010110001000101100000  0.0667  0.0679  0.99915
 2 0001000100010010011000101000000  0.0667  0.1924  0.00564
 3 0001000100110010001000101000000  0.0672  0.0674  0.99967
 4 0001000101010010010000101000000  0.0677  0.1299  0.02218
 5 0001000000110111000000000110000  0.0633  0.0015  0.01517

Average Values:                        0.0663  0.0918  0.40836

Average Function Value of Generation= 0.40836
Maximum Function Value               = 0.99967


 Number of Crossovers        =   67


################# Generation  112  #################
 #      Binary Code                    Param1  Param2  Fitness
 1 0001000100010110001000101100000  0.0667  0.0679  0.99915
 2 0001000100110010001000101000000  0.0672  0.0674  0.99967
 3 0001000100010110001000101000000  0.0667  0.0674  0.99976
 4 0001000100010010000000101100000  0.0667  0.0054  0.02816
 5 0001000000110111000100000100000  0.0633  0.0635  0.98193

Average Values:                        0.0661  0.0543  0.80174

Average Function Value of Generation= 0.80174
Maximum Function Value               = 0.99976


 Number of Crossovers        =   72
 Elitist Reproduction on Individual    2
```

```
#################  Generation  113  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 000100000011011000100010100000  0.0633  0.0674  0.99029
  2 000100010001011000100010100000  0.0667  0.0674  0.99976
  3 000100010011011000100010100000  0.0672  0.0674  0.99964
  4 000100010011001000100010100000  0.0672  0.0674  0.99967
  5 000100010011011000100010100000  0.0672  0.0674  0.99964

 Average Values:                    0.0663  0.0674  0.99780

 Average Function Value of Generation= 0.99780
 Maximum Function Value              = 0.99976


  Number of Crossovers      =   91


%%%%%%%  Restart micro-population at generation  113  %%%%%%%


#################  Generation  114  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 000100010001011000100010100000  0.0667  0.0674  0.99976
  2 000010011101100011000100111100  0.0385  0.3847  0.00088
  3 110110111010100101011011011011  0.8581  0.6786  0.00816
  4 010110001101110011000001011011  0.3471  0.3778  0.00000
  5 011011010011110100001010100111  0.4267  0.5207  0.00276

 Average Values:                    0.3474  0.4058  0.20231

 Average Function Value of Generation= 0.20231
 Maximum Function Value              = 0.99976


  Number of Crossovers      =   64
  Elitist Reproduction on Individual    4


#################  Generation  115  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 111010110010110101001010101111  0.9187  0.6460  0.00010
  2 011010010011010100001010100011  0.4110  0.5206  0.00094
  3 010010110010100100001011011111  0.2936  0.5224  0.00324
  4 000100010001011000100010100000  0.0667  0.0674  0.99976
  5 000110011001101010100000111100  0.1000  0.3143  0.03076

 Average Values:                    0.3580  0.4142  0.20696

 Average Function Value of Generation= 0.20696
 Maximum Function Value              = 0.99976


  Number of Crossovers      =   75
  Elitist Reproduction on Individual    5


#################  Generation  116  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 000110011001001000100000100000  0.0999  0.0635  0.40759
  2 000000010001100100101010100100  0.0043  0.5831  0.00003
  3 001100010001011100101010100001  0.1917  0.5831  0.00001
  4 000000010001001100101011011110  0.0042  0.5849  0.00005
  5 000100010001011000100010100000  0.0667  0.0674  0.99976
```

```
Average Values:                          0.0734  0.3764  0.28149

Average Function Value of Generation= 0.28149
Maximum Function Value               = 0.99976


 Number of Crossovers        =   76
 Elitist Reproduction on Individual    1


#################  Generation  117  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 00010001000101100010001010000 0.0667  0.0674  0.99976
  2 00010001000101000010101010000 0.0667  0.0830  0.81463
  3 00001001000110010010001010000 0.0355  0.5674  0.00000
  4 00010001100100100010001010000 0.0686  0.0674  0.99724
  5 00010001000100100010000010000 0.0667  0.0635  0.99131

 Average Values:                         0.0609  0.1697  0.76059

 Average Function Value of Generation= 0.76059
 Maximum Function Value               = 0.99976


  Number of Crossovers        =   81
  Elitist Reproduction on Individual    3

#################  Generation  118  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 00010001000100000010101010000 0.0667  0.0830  0.81462
  2 00010001100101100010001010000 0.0687  0.0674  0.99707
  3 00010001000101100010001010000 0.0667  0.0674  0.99976
  4 00010001000100100010000010000 0.0667  0.0635  0.99131
  5 00010001000100100010001010000 0.0667  0.0674  0.99975

 Average Values:                         0.0671  0.0697  0.96050

 Average Function Value of Generation= 0.96050
 Maximum Function Value               = 0.99976


  Number of Crossovers        =   83
  Elitist Reproduction on Individual    5


%%%%%%%  Restart micro-population at generation  118  %%%%%%%


#################  Generation  119  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 00010001000101100010001010000 0.0667  0.0674  0.99976
  2 01110110101010000000010000101 0.4635  0.0041  0.01153
  3 11111111110001110100110001110 1 0.9991  0.6493  0.00071
  4 10111010001010110100010001110 0 0.7272  0.6337  0.00011
  5 11001011001111100100111001110 1 0.7939  0.1532  0.00000

 Average Values:                         0.6101  0.3015  0.20242

 Average Function Value of Generation= 0.20242
 Maximum Function Value               = 0.99976


  Number of Crossovers        =   62
```

```
Elitist Reproduction on Individual    4


#################  Generation  120  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 001110010100011000001100010101  0.2237  0.0241  0.05406
  2 011111111010011001001100001101  0.4986  0.1488  0.00003
  3 010100011100011100101000111001  0.3194  0.5799  0.00003
  4 000100010001011000100010100000  0.0667  0.0674  0.99976
  5 110110110001011001101100011001  0.8558  0.2117  0.00608

 Average Values:                    0.3929  0.2064  0.21199

 Average Function Value of Generation= 0.21199
 Maximum Function Value             = 0.99976


  Number of Crossovers        =   80
  Elitist Reproduction on Individual    2


#################  Generation  121  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 001110010101011000001010010001  0.2240  0.0201  0.03914
  2 000100010001011000100010100000  0.0667  0.0674  0.99976
  3 001100010001011000101000000001  0.1917  0.0782  0.00447
  4 001100010101011000100000110100  0.1927  0.0641  0.00600
  5 101110110000011000001100011001  0.7306  0.0242  0.00006

 Average Values:                    0.2812  0.0508  0.20989

 Average Function Value of Generation= 0.20989
 Maximum Function Value             = 0.99976


  Number of Crossovers        =   68
  Elitist Reproduction on Individual    5


#################  Generation  122  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 001110010101011000001010010100  0.2240  0.0201  0.03946
  2 001100010001011000100010110100  0.1917  0.0680  0.00494
  3 001110010001011000100010100000  0.2230  0.0674  0.24005
  4 000100010101011000100010100000  0.0677  0.0674  0.99915
  5 000100010001011000100010100000  0.0667  0.0674  0.99976

 Average Values:                    0.1546  0.0581  0.45667

 Average Function Value of Generation= 0.45667
 Maximum Function Value             = 0.99976


  Number of Crossovers        =   62


#################  Generation  123  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001011000100010100000  0.0667  0.0674  0.99976
  2 000100010101011000100010100000  0.0677  0.0674  0.99915
  3 001110010101011000100010100000  0.2240  0.0674  0.25678
  4 000100010001011000100010100000  0.0667  0.0674  0.99976
  5 000100010101011000101010010100  0.0677  0.0826  0.82173
```

Average Values: 0.0986 0.0704 0.81543

Average Function Value of Generation= 0.81543
Maximum Function Value = 0.99976

Number of Crossovers = 65

%%%%%%% Restart micro-population at generation 123 %%%%%%%

################# Generation 124 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001011000100010100000	0.0667	0.0674	0.99976
2	110100101000000001011101000011	0.8223	0.1817	0.00001
3	000110111101000101011000011111	0.1086	0.6728	0.03662
4	010100100010001100010001101010	0.3208	0.5345	0.00003
5	011100110100111000101110010111	0.4504	0.0905	0.31949

Average Values: 0.3538 0.3094 0.27118

Average Function Value of Generation= 0.27118
Maximum Function Value = 0.99976

Number of Crossovers = 82
Elitist Reproduction on Individual 1

################# Generation 125 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001011000100010100000	0.0667	0.0674	0.99976
2	001100111100111101001110011111	0.2024	0.6533	0.00677
3	001100110100011000101010010001	0.2003	0.0826	0.01840
4	000110110101011100001010101110	0.1068	0.5210	0.00288
5	000100010001111000101110000011	0.0669	0.0899	0.65492

Average Values: 0.1286 0.2828 0.33655

Average Function Value of Generation= 0.33655
Maximum Function Value = 0.99976

Number of Crossovers = 64
Elitist Reproduction on Individual 5

################# Generation 126 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001111000101010100000	0.0669	0.0830	0.81464
2	000100110101011000100010110001	0.0755	0.0679	0.94203
3	000100010001011000101010000010	0.0667	0.0821	0.83340
4	000100010001111000100110000011	0.0669	0.0743	0.95748
5	000100010001011000100010100000	0.0667	0.0674	0.99976

Average Values: 0.0686 0.0749 0.90946

Average Function Value of Generation= 0.90946
Maximum Function Value = 0.99976

Number of Crossovers = 70
Elitist Reproduction on Individual 2

```
################ Generation  127  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010001111000100010100010  0.0669  0.0674  0.99971
 2 000100010001011000100010100000  0.0667  0.0674  0.99976
 3 000100010001011000101010100010  0.0667  0.0831  0.81336
 4 000100010001011000100110100001  0.0667  0.0752  0.94670
 5 000100010001011000100010100010  0.0667  0.0674  0.99970

Average Values:                    0.0668  0.0721  0.95184

Average Function Value of Generation= 0.95184
Maximum Function Value              = 0.99976


 Number of Crossovers      =   69


%%%%%%%  Restart micro-population at generation  127  %%%%%%%


################ Generation  128  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010001011000100010100000  0.0667  0.0674  0.99976
 2 010100011111100100111011010001  0.3202  0.6158  0.00287
 3 010101111110111001111101111001  0.3435  0.2459  0.00022
 4 001011111011011001011011110101  0.1864  0.1794  0.00000
 5 010110100010100111010101101100  0.3522  0.9174  0.00000

Average Values:                    0.2538  0.4052  0.20057

Average Function Value of Generation= 0.20057
Maximum Function Value              = 0.99976


 Number of Crossovers      =   80
 Elitist Reproduction on Individual    4

################ Generation  129  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000100011001111000110011000000  0.0688  0.0996  0.41629
 2 000001111001011001110011100001  0.0296  0.2256  0.09144
 3 000101110101011000111111101001  0.0912  0.1243  0.03026
 4 000100010001011000100010100000  0.0667  0.0674  0.99976
 5 000100011111111000111011010001  0.0703  0.2408  0.59074

Average Values:                    0.0653  0.1515  0.42570

Average Function Value of Generation= 0.42570
Maximum Function Value              = 0.99976


 Number of Crossovers      =   72
 Elitist Reproduction on Individual    4


################ Generation  130  ################
 #       Binary Code                Param1  Param2  Fitness
 1 000100010111011001100010010001  0.0682  0.1919  0.00513
 2 000100010001110001101010110001  0.0668  0.2085  0.06531
 3 000100010001010001110010010000  0.0667  0.2232  0.24269
 4 000100010001011000100010100000  0.0667  0.0674  0.99976
```

5	000100011001011000110010100000	0.0687	0.0986	0.43942

Average Values: 0.0674 0.1579 0.35046

Average Function Value of Generation= 0.35046
Maximum Function Value = 0.99976

Number of Crossovers = 76
Elitist Reproduction on Individual 3

################# Generation 131 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001010000100010010000	0.0667	0.0669	0.99999
2	000100010001010000100010100000	0.0667	0.0674	0.99975
3	000100010001011000100010100000	0.0667	0.0674	0.99976
4	000100010001011001110010110000	0.0667	0.2241	0.25951
5	000100011001011000110010100000	0.0687	0.0986	0.43942

Average Values: 0.0671 0.1049 0.73968

Average Function Value of Generation= 0.73968
Maximum Function Value = 0.99999

Number of Crossovers = 77

################# Generation 132 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001010000100010100000	0.0667	0.0674	0.99975
2	000100010001010000100010010000	0.0667	0.0669	0.99999
3	000100011001011000110010000000	0.0687	0.0977	0.46276
4	000100010001011000100010100000	0.0667	0.0674	0.99976
5	000100010001010000100010010000	0.0667	0.0669	0.99999

Average Values: 0.0671 0.0732 0.89245

Average Function Value of Generation= 0.89245
Maximum Function Value = 0.99999

Number of Crossovers = 75

################# Generation 133 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001010000100010010000	0.0667	0.0669	0.99999
2	000100010001010000100010010000	0.0667	0.0669	0.99999
3	000100010001010000100010000000	0.0667	0.0664	0.99985
4	000100010001011000100010100000	0.0667	0.0674	0.99976
5	000100010001010000100010100000	0.0667	0.0674	0.99975

Average Values: 0.0667 0.0670 0.99987

Average Function Value of Generation= 0.99987
Maximum Function Value = 0.99999

Number of Crossovers = 78

%%%%%%% Restart micro-population at generation 133 %%%%%%%

```
#################  Generation  134  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 00010001000101000010001001000 0      0.0667  0.0669  0.99999
  2 11111000101010000100010011010 0      0.9713  0.1344  0.00000
  3 01001010001111000001001101110 0      0.2900  0.0380  0.23019
  4 01010001110111111101001010101 1      0.3198  0.9115  0.00006
  5 00110000100100101111101000000 0      0.1897  0.4883  0.00074

 Average Values:                         0.3675  0.3278  0.24620

 Average Function Value of Generation= 0.24620
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   89
  Elitist Reproduction on Individual    2


#################  Generation  135  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 000100010001010000110010011100       0.0667  0.0985  0.44350
  2 000100010001010000100010010000       0.0667  0.0669  0.99999
  3 001100000001000001110010010000       0.1877  0.2232  0.00048
  4 010100011101010111010010101000       0.3197  0.9114  0.00006
  5 000000110001010000100010010000       0.0120  0.0669  0.06694

 Average Values:                         0.1306  0.2734  0.30219

 Average Function Value of Generation= 0.30219
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   74
  Elitist Reproduction on Individual    2


#################  Generation  136  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 000000010001010000100010011000       0.0042  0.0671  0.02371
  2 000100010001010000100010010000       0.0667  0.0669  0.99999
  3 000100010001010000100010011100       0.0667  0.0673  0.99985
  4 000000110001010000110010010000       0.0120  0.0981  0.03027
  5 000100010001010000110010010100       0.0667  0.0983  0.44934

 Average Values:                         0.0433  0.0795  0.50063

 Average Function Value of Generation= 0.50063
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   72


#################  Generation  137  #################
  #       Binary Code                    Param1  Param2  Fitness
  1 000100010001010000100010010000       0.0667  0.0669  0.99999
  2 000100010001010000100010011100       0.0667  0.0673  0.99985
  3 000100010001010000100010011000       0.0667  0.0671  0.99992
  4 000100010001010000100010010100       0.0667  0.0670  0.99996
  5 000100010001010000100010011100       0.0667  0.0673  0.99985

 Average Values:                         0.0667  0.0671  0.99991
```

```
 Average Function Value of Generation= 0.99991
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   76


%%%%%%%  Restart micro-population at generation  137  %%%%%%%


################  Generation  138  ################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010001010000100010010000  0.0667  0.0669  0.99999
  2 101001000111011110111011011111  0.6424  0.8662  0.01105
  3 010000110101001010011011111100  0.2630  0.3046  0.15539
  4 101101010011011111001101110100  0.7079  0.9020  0.00007
  5 111111110000111111011000101100  0.9963  0.9232  0.00000

 Average Values:                    0.5353  0.6126  0.23330

 Average Function Value of Generation= 0.23330
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    4


################  Generation  139  ################
  #       Binary Code                Param1  Param2  Fitness
  1 001100000101011110110011010110  0.1888  0.8503  0.00018
  2 010100010001001010000011110100  0.3167  0.2575  0.04855
  3 001001000111010100100010010100  0.1424  0.5670  0.00000
  4 000100010001010000100010010000  0.0667  0.0669  0.99999
  5 000100010101010000000101011000  0.0677  0.0210  0.16562

 Average Values:                    0.1565  0.3525  0.24287

 Average Function Value of Generation= 0.24287
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   75
  Elitist Reproduction on Individual    4


################  Generation  140  ################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010101010000000101011000  0.0677  0.0210  0.16562
  2 001100010101011000010010010010  0.1927  0.0357  0.00276
  3 000100010001010000101010110000  0.0667  0.0835  0.80434
  4 000100010001010000100010010000  0.0667  0.0669  0.99999
  5 010100010001011010000011110100  0.3168  0.2575  0.04821

 Average Values:                    0.1421  0.0929  0.40418

 Average Function Value of Generation= 0.40418
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   67
```

```
#################  Generation  141  #################
  #       Binary Code                 Param1   Param2   Fitness
  1 000100010001011010000011110100  0.0667   0.2575   0.83477
  2 000100010001010000101010110000  0.0667   0.0835   0.80434
  3 000100010001010000100010010000  0.0667   0.0669   0.99999
  4 000100010001010000100010010000  0.0667   0.0669   0.99999
  5 000100010001010000101010110000  0.0667   0.0835   0.80434

 Average Values:                      0.0667   0.1116   0.88868

 Average Function Value of Generation= 0.88868
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   81


#################  Generation  142  #################
  #       Binary Code                 Param1   Param2   Fitness
  1 000100010001010000000011110100  0.0667   0.0074   0.03768
  2 000100010001010010100010010100  0.0667   0.3170   0.05596
  3 000100010001010000101010010000  0.0667   0.0825   0.82473
  4 000100010001011000100011110100  0.0667   0.0699   0.99250
  5 000100010001010000100010010000  0.0667   0.0669   0.99999

 Average Values:                      0.0667   0.1088   0.58217

 Average Function Value of Generation= 0.58217
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   65


#################  Generation  143  #################
  #       Binary Code                 Param1   Param2   Fitness
  1 000100010001010000100010010000  0.0667   0.0669   0.99999
  2 000100010001011000100010010100  0.0667   0.0670   0.99997
  3 000100010001011000100011110000  0.0667   0.0698   0.99307
  4 000100010001010000101010110000  0.0667   0.0835   0.80434
  5 000100010001010000100010010100  0.0667   0.0670   0.99996

 Average Values:                      0.0667   0.0709   0.95946

 Average Function Value of Generation= 0.95946
 Maximum Function Value              = 0.99999


  Number of Crossovers      =   66


%%%%%%%  Restart micro-population at generation  143  %%%%%%%


#################  Generation  144  #################
  #       Binary Code                 Param1   Param2   Fitness
  1 000100010001010000100010010000  0.0667   0.0669   0.99999
  2 110001001101000011011011010101  0.7688   0.4284   0.00001
  3 100110000101111010001011101101  0.5952   0.2729   0.00823
  4 101000110111001101110100011000  0.6385   0.7273   0.00012
  5 011011111011011101101101000011  0.4364   0.7130   0.00280

 Average Values:                      0.5011   0.4417   0.20223
```

```
 Average Function Value of Generation= 0.20223
 Maximum Function Value              = 0.99999


  Number of Crossovers        =   80
  Elitist Reproduction on Individual    1


################  Generation  145  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001010000100010010000  0.0667  0.0669  0.99999
  2 010011110001010100100011000010  0.3089  0.5684  0.00000
  3 100100010101111010001011001001  0.5679  0.2718  0.00001
  4 000110010001011000100010110101  0.0980  0.0680  0.45542
  5 010011000011011010001001000101  0.2977  0.2677  0.24427

 Average Values:                    0.2678  0.2486  0.33994

 Average Function Value of Generation= 0.33994
 Maximum Function Value              = 0.99999


  Number of Crossovers        =   66
  Elitist Reproduction on Individual    1


################  Generation  146  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001010000100010010000  0.0667  0.0669  0.99999
  2 000011010011011010000010000001  0.0516  0.2539  0.67346
  3 010111000011011000001011010101  0.3602  0.0221  0.00000
  4 000100010001010000100010010101  0.0667  0.0670  0.99995
  5 010100000011011010001011000000  0.3133  0.2715  0.06591

 Average Values:                    0.1717  0.1363  0.54786

 Average Function Value of Generation= 0.54786
 Maximum Function Value              = 0.99999


  Number of Crossovers        =   58
  Elitist Reproduction on Individual    4


################  Generation  147  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000101010011010000000010010101  0.0828  0.0045  0.02041
  2 000110010011011000100010000000  0.0985  0.0664  0.44417
  3 000100010001010000100010010001  0.0667  0.0669  0.99998
  4 000100010001010000100010010000  0.0667  0.0669  0.99999
  5 000100010011010010101011000000  0.0672  0.3340  0.00395

 Average Values:                    0.0764  0.1078  0.49370

 Average Function Value of Generation= 0.49370
 Maximum Function Value              = 0.99999


  Number of Crossovers        =   76
  Elitist Reproduction on Individual    5


################  Generation  148  ################
  #      Binary Code                  Param1  Param2  Fitness
```

```
 1 000101010001010000100010010101  0.0823  0.0670  0.82844
 2 000110010001011000100010010000  0.0980  0.0669  0.45592
 3 000100010001011000100010010000  0.0667  0.0669  0.99999
 4 000100010001011000100010000000  0.0667  0.0664  0.99986
 5 000100010001010000100010010000  0.0667  0.0669  0.99999

Average Values:                   0.0761  0.0668  0.85684

Average Function Value of Generation= 0.85684
Maximum Function Value             = 0.99999


 Number of Crossovers       =   74


%%%%%%%  Restart micro-population at generation  148  %%%%%%%


#################  Generation  149  #################
 #       Binary Code                 Param1  Param2  Fitness
 1 000100010001011000100010010000  0.0667  0.0669  0.99999
 2 100011010010011101101111011111  0.5514  0.7178  0.00000
 3 111111000011101010001110101110  0.9853  0.2788  0.00047
 4 000101000010001011011101000100  0.0786  0.4318  0.27929
 5 001000110000001111100011100011  0.1368  0.9445  0.00000

Average Values:                   0.3638  0.4879  0.25595

Average Function Value of Generation= 0.25595
Maximum Function Value             = 0.99999


 Number of Crossovers       =   74
 Elitist Reproduction on Individual    5


#################  Generation  150  #################
 #       Binary Code                 Param1  Param2  Fitness
 1 100011010011011100101010010100  0.5516  0.5827  0.00000
 2 000101000000001001100011000100  0.0782  0.1935  0.00634
 3 000100000000011000101011010000  0.0626  0.0845  0.77239
 4 100100000011001000001010100100  0.5633  0.0206  0.00000
 5 000100010001011000100010010000  0.0667  0.0669  0.99999

Average Values:                   0.2645  0.1896  0.35574

Average Function Value of Generation= 0.35574
Maximum Function Value             = 0.99999


 Number of Crossovers       =   69
 Elitist Reproduction on Individual    1


#################  Generation  151  #################
 #       Binary Code                 Param1  Param2  Fitness
 1 000100010001011000100010010000  0.0667  0.0669  0.99999
 2 000100000001011000100010010100  0.0628  0.0670  0.98769
 3 000101000000011000101011010000  0.0782  0.0845  0.70799
 4 000101010000011001100010010000  0.0821  0.1919  0.00425
 5 100100000001011000100010000100  0.5629  0.0665  0.00000

Average Values:                   0.1706  0.0954  0.53998
```

```
Average Function Value of Generation= 0.53998
Maximum Function Value              = 0.99999


 Number of Crossovers       =   77
 Elitist Reproduction on Individual    2


################# Generation  152  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000100010000011001100010010000  0.0665  0.1919  0.00510
  2 000100010001011000100010010000  0.0667  0.0669  0.99999
  3 000100010001011000100010010100  0.0667  0.0670  0.99997
  4 000100010001011000100010010100  0.0667  0.0670  0.99997
  5 000101010001011001100010010000  0.0824  0.1919  0.00422

 Average Values:                     0.0698  0.1169  0.60185

 Average Function Value of Generation= 0.60185
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   75

%%%%%%  Restart micro-population at generation  152  %%%%%%

################# Generation  153  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 000100010001011000100010010000  0.0667  0.0669  0.99999
  2 011101010010000010110011000111  0.4575  0.3498  0.00001
  3 010010101001100010100000111100  0.2914  0.3143  0.03167
  4 100100100101011001101101000001  0.5716  0.2129  0.00001
  5 110101011000100010010101111110  0.8341  0.2929  0.02422

 Average Values:                     0.4443  0.2474  0.21118

 Average Function Value of Generation= 0.21118
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    4


################# Generation  154  #################
  #      Binary Code                 Param1  Param2  Fitness
  1 110100010001001000010111011100  0.8167  0.0458  0.02350
  2 010010110001111000100010110000  0.2934  0.0679  0.37844
  3 010100010000001000110011000001  0.3164  0.0996  0.02493
  4 000100010001011000100010010000  0.0667  0.0669  0.99999
  5 110001101001100010110000111100  0.7758  0.3456  0.00000

 Average Values:                     0.4538  0.1252  0.28537

 Average Function Value of Generation= 0.28537
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   68
  Elitist Reproduction on Individual    2
```

```
################  Generation  155  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001011000110011000000   0.0667  0.0996  0.41756
  2 000100010001011000100010010000   0.0667  0.0669  0.99999
  3 110110110001001000000110111100   0.8558  0.0136  0.00528
  4 010110010001011000100010010000   0.3480  0.0669  0.00005
  5 010000010000111000100011010000   0.2541  0.0688  0.80658

 Average Values:                     0.3183  0.0632  0.44589

 Average Function Value of Generation= 0.44589
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   70
  Elitist Reproduction on Individual    1


################  Generation  156  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001011000100010010000   0.0667  0.0669  0.99999
  2 010000010000001000000110010100   0.2539  0.0123  0.05599
  3 000100010001011000110010000000   0.0667  0.0977  0.46401
  4 010110110001001000100110111000   0.3558  0.0759  0.00000
  5 000100010001011000110010010000   0.0667  0.0981  0.45226

 Average Values:                     0.1620  0.0702  0.39445

 Average Function Value of Generation= 0.39445
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   76
  Elitist Reproduction on Individual    5

################  Generation  157  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001001000110110010100   0.0667  0.1061  0.27783
  2 000000010001011000100110010000   0.0042  0.0747  0.02270
  3 000100010001011000110010000000   0.0667  0.0977  0.46401
  4 000000010001011000100010010100   0.0042  0.0670  0.02382
  5 000100010001011000100010010000   0.0667  0.0669  0.99999

 Average Values:                     0.0417  0.0825  0.35767

 Average Function Value of Generation= 0.35767
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    3


################  Generation  158  ################
  #      Binary Code                  Param1  Param2  Fitness
  1 000100010001011000110010010000   0.0667  0.0981  0.45226
  2 000100010001001000110010000000   0.0667  0.0977  0.46400
  3 000100010001011000100010010000   0.0667  0.0669  0.99999
  4 000000010001011000100010010000   0.0042  0.0669  0.02382
  5 000100010001011000110010000000   0.0667  0.0977  0.46401

 Average Values:                     0.0542  0.0855  0.48082
```

```
Average Function Value of Generation= 0.48082
Maximum Function Value              = 0.99999


 Number of Crossovers       =    78


%%%%%%%  Restart micro-population at generation  158  %%%%%%%


#################  Generation  159  #################
  #       Binary Code                  Param1  Param2  Fitness
  1 00010001000101100010001001 0000    0.0667  0.0669  0.99999
```

#	Binary Code	Param1	Param2	Fitness
1	00010001000101100010001001 0000			

```
```

```
 #       Binary Code                    Param1  Param2  Fitness
 1 00010010000101100010001001000 0      0.0707  0.0669  0.98876
 2 000101100001011000100010010000       0.0863  0.0669  0.74257
 3 000100010001011000100010010000       0.0667  0.0669  0.99999
 4 011101100001011000000110010000       0.4613  0.0122  0.03469
 5 000100010000011000000010000000       0.0665  0.0039  0.02263

Average Values:                         0.1503  0.0434  0.55773

Average Function Value of Generation= 0.55773
Maximum Function Value              = 0.99999


 Number of Crossovers       =   77
 Elitist Reproduction on Individual    5


################# Generation  163  #################
 #       Binary Code                    Param1  Param2  Fitness
 1 000100000001011000100010010000       0.0628  0.0669  0.98771
 2 000100110001011000100010010000       0.0746  0.0669  0.95472
 3 010101100001011000000110010000       0.3363  0.0122  0.00016
 4 000100100001011000100010010000       0.0707  0.0669  0.98876
 5 000100010001011000100010010000       0.0667  0.0669  0.99999

Average Values:                         0.1222  0.0560  0.78627

Average Function Value of Generation= 0.78627
Maximum Function Value              = 0.99999


 Number of Crossovers       =   84

%%%%%%%  Restart micro-population at generation  163  %%%%%%%

################# Generation  164  #################
 #       Binary Code                    Param1  Param2  Fitness
 1 000100010001011000100010010000       0.0667  0.0669  0.99999
 2 111000110101011111111101001111       0.8881  0.9946  0.00004
 3 110111101110101010110011010111       0.8708  0.3503  0.00000
 4 100011110000011110111000101110       0.5587  0.8608  0.00000
 5 110010011000110111101110110111       0.7873  0.9666  0.00000

Average Values:                         0.6343  0.6478  0.20001

Average Function Value of Generation= 0.20001
Maximum Function Value              = 0.99999


 Number of Crossovers       =   82
 Elitist Reproduction on Individual    5


################# Generation  165  #################
 #       Binary Code                    Param1  Param2  Fitness
 1 111001101111001111110011001111       0.9022  0.9751  0.00000
 2 101100010001011000101111010100       0.6918  0.0924  0.03626
 3 110100111111001000110010010000       0.8279  0.0981  0.02402
 4 000100000000111010100011010001       0.0627  0.3189  0.04426
 5 000100010001011000100010010000       0.0667  0.0669  0.99999

Average Values:                         0.5103  0.3103  0.22091
```

```
 Average Function Value of Generation= 0.22091
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   67
  Elitist Reproduction on Individual    2


################# Generation  166  #################
  #       Binary Code                Param1  Param2  Fitness
  1 00010001000001101010001001001  0.0665  0.3169  0.05655
  2 000100010001011000100010010000  0.0667  0.0669  0.99999
  3 000100010000111010100011010000  0.0666  0.3189  0.04501
  4 001100010000111000100111010001  0.1916  0.0767  0.00447
  5 000100111001011000100010010000  0.0765  0.0669  0.92975

 Average Values:                     0.0936  0.1693  0.40715

 Average Function Value of Generation= 0.40715
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    5


################# Generation  167  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100110001011000100010010000  0.0746  0.0669  0.95472
  2 000100010000011010100010010001  0.0665  0.3169  0.05655
  3 000100110001011000100010010000  0.0746  0.0669  0.95472
  4 000100010000111000100010010000  0.0666  0.0669  0.99996
  5 000100010001011000100010010000  0.0667  0.0669  0.99999

 Average Values:                     0.0698  0.1169  0.79319

 Average Function Value of Generation= 0.79319
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   83


%%%%%%%  Restart micro-population at generation  167  %%%%%%%


################# Generation  168  #################
  #       Binary Code                Param1  Param2  Fitness
  1 000100010001011000100010010000  0.0667  0.0669  0.99999
  2 010000011111001000100100111010  0.2576  0.0721  0.81793
  3 100011110010101010110001101011  0.5593  0.3470  0.00000
  4 101110001000100001010110101101  0.7208  0.1693  0.00000
  5 011101100110011010011111110100  0.4625  0.3121  0.04740

 Average Values:                     0.4134  0.1935  0.37306

 Average Function Value of Generation= 0.37306
 Maximum Function Value              = 0.99999


  Number of Crossovers       =   80
  Elitist Reproduction on Individual    5
```

################# Generation 169 #################

#	Binary Code	Param1	Param2	Fitness
1	0111011111100110100001011100000	0.4684	0.2612	0.39336
2	0011010101000110101010100100000	0.2081	0.3325	0.00033
3	0100000110110010001000000010010	0.2566	0.0631	0.82055
4	0011100100000000000100010111101	0.2227	0.0683	0.23412
5	0001000100010110001000100100000	0.0667	0.0669	0.99999
Average Values:		0.2445	0.1584	0.48967

Average Function Value of Generation= 0.48967
Maximum Function Value = 0.99999

Number of Crossovers = 88
Elitist Reproduction on Individual 3

################# Generation 170 #################

#	Binary Code	Param1	Param2	Fitness
1	0101010100010110000000011110000	0.3324	0.0073	0.00020
2	0101000110010010001000000010000	0.3186	0.0630	0.04567
3	0001000100010110001000100100000	0.0667	0.0669	0.99999
4	0111001111100110001000001110000	0.4527	0.0659	0.50846
5	0011100100000000000100010011100	0.2227	0.0673	0.23446
Average Values:		0.2786	0.0541	0.35776

Average Function Value of Generation= 0.35776
Maximum Function Value = 0.99999

Number of Crossovers = 70
Elitist Reproduction on Individual 1

################# Generation 171 #################

#	Binary Code	Param1	Param2	Fitness
1	0001000100010110001000100100000	0.0667	0.0669	0.99999
2	0011000100010110001000100100000	0.1917	0.0669	0.00494
3	0101000111110110001000111110000	0.3202	0.0698	0.03793
4	0101000110010110001000100100000	0.3187	0.0669	0.04586
5	0111000101000000000100010111000	0.4424	0.0681	0.43721
Average Values:		0.2680	0.0677	0.30519

Average Function Value of Generation= 0.30519
Maximum Function Value = 0.99999

Number of Crossovers = 80
Elitist Reproduction on Individual 4

################# Generation 172 #################

#	Binary Code	Param1	Param2	Fitness
1	0011000100000100001000101110000	0.1915	0.0681	0.00466
2	0111000110000000000100010110000	0.4434	0.0679	0.44695
3	0001000101110110001000111110000	0.0682	0.0698	0.99162
4	0001000100010110001000100100000	0.0667	0.0669	0.99999
5	0001000110110110001000101100000	0.0692	0.0679	0.99488
Average Values:		0.1678	0.0681	0.68762

```
 Average Function Value of Generation= 0.68762
 Maximum Function Value               = 0.99999


  Number of Crossovers       =   72
  Elitist Reproduction on Individual    4


################# Generation  173  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010111011000100010110000  0.0682  0.0679  0.99769
  2 000100010111011000100011010000  0.0682  0.0688  0.99539
  3 000100011001011000100010110000  0.0687  0.0679  0.99647
  4 000100010001011000100010010000  0.0667  0.0669  0.99999
  5 000100010111011000100010010000  0.0682  0.0669  0.99853

 Average Values:                      0.0680  0.0677  0.99761

 Average Function Value of Generation= 0.99761
 Maximum Function Value               = 0.99999


  Number of Crossovers       =   73


################# Generation  174  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010011011000100010010000  0.0672  0.0669  0.99987
  2 000100010001011000100010010000  0.0667  0.0669  0.99999
  3 000100010011011000100010010000  0.0672  0.0669  0.99987
  4 000100011111011000100010110000  0.0702  0.0679  0.99061
  5 000100011001011000100010010000  0.0687  0.0669  0.99730

 Average Values:                      0.0680  0.0671  0.99753

 Average Function Value of Generation= 0.99753
 Maximum Function Value               = 0.99999


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    3


%%%%%%%  Restart micro-population at generation  174  %%%%%%%


################# Generation  175  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001011000100010010000  0.0667  0.0669  0.99999
  2 100100000010001101100111100111  0.5630  0.7024  0.00000
  3 000001100000010010000110001010  0.0235  0.2620  0.17312
  4 101111001101110001001010101001  0.7378  0.1458  0.00000
  5 010100111100100000001000101111  0.3273  0.0171  0.00157

 Average Values:                      0.3437  0.2388  0.23494

 Average Function Value of Generation= 0.23494
 Maximum Function Value               = 0.99999


  Number of Crossovers       =   78
  Elitist Reproduction on Individual    5
```

################# Generation 176 #################

#	Binary Code	Param1	Param2	Fitness
1	101101000001011000001010010001	0.7035	0.0201	0.00348
2	000001000000001000000110011000	0.0157	0.0125	0.00703
3	010001100100110010001000101011	0.2746	0.2669	0.61912
4	010100111101111000000000010110	0.3276	0.0007	0.00017
5	000100010001011000100010010000	0.0667	0.0669	0.99999
Average Values:		0.2776	0.0734	0.32596

Average Function Value of Generation= 0.32596
Maximum Function Value = 0.99999

Number of Crossovers = 78
Elitist Reproduction on Individual 4

################# Generation 177 #################

#	Binary Code	Param1	Param2	Fitness
1	010001110000110010101010001000	0.2775	0.3323	0.00390
2	000101010001010000100110010000	0.0823	0.0747	0.78949
3	000001000000110010001000001011	0.0158	0.2660	0.08468
4	000100010001011000100010010000	0.0667	0.0669	0.99999
5	000001000000001000000111010101011	0.0157	0.0287	0.02993
Average Values:		0.0916	0.1537	0.38160

Average Function Value of Generation= 0.38160
Maximum Function Value = 0.99999

Number of Crossovers = 79
Elitist Reproduction on Individual 1

################# Generation 178 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001011000100010010000	0.0667	0.0669	0.99999
2	000101010001011000100110010000	0.0824	0.0747	0.78890
3	000101010001011000100010010000	0.0824	0.0669	0.82785
4	000100010001011000100110010000	0.0667	0.0747	0.95295
5	000001010001010000100110011000	0.0198	0.0750	0.14202
Average Values:		0.0636	0.0716	0.74234

Average Function Value of Generation= 0.74234
Maximum Function Value = 0.99999

Number of Crossovers = 90

%%%%%%% Restart micro-population at generation 178 %%%%%%%

################# Generation 179 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001011000100010010000	0.0667	0.0669	0.99999
2	101110110010011100101100100010	0.7311	0.5870	0.00000
3	111100011111101011011110111101	0.9452	0.4355	0.00000
4	001100011110110111011111001110	0.1950	0.9360	0.00000
5	101011111000010001101000010101	0.6856	0.2038	0.00328

Average Values: 0.5247 0.4458 0.20065

Average Function Value of Generation= 0.20065
Maximum Function Value = 0.99999

Number of Crossovers = 77
Elitist Reproduction on Individual 1

################# Generation 180 #################

#	Binary Code	Param1	Param2	Fitness
1	000100010001011000100010010000	0.0667	0.0669	0.99999
2	000110011000011001101000010101	0.0997	0.2038	0.01514
3	101011110000011000101100110010	0.6837	0.0875	0.07134
4	001100011101110100111111000010	0.1948	0.6231	0.00103
5	100110010011011100101110000000	0.5985	0.5899	0.00008
Average Values:		0.3287	0.3142	0.21752

Average Function Value of Generation= 0.21752
Maximum Function Value = 0.99999

Number of Crossovers = 81
Elitist Reproduction on Individual 4

################# Generation 181 #################

#	Binary Code	Param1	Param2	Fitness
1	101110110000011000100110010000	0.7306	0.0747	0.00027
2	101111010000011000101100010101	0.7384	0.0866	0.00002
3	101100110001011000100110010000	0.6996	0.0747	0.03146
4	000100010001011000100010010000	0.0667	0.0669	0.99999
5	001011111000011001101100010111	0.1856	0.2116	0.00010
Average Values:		0.4842	0.1029	0.20637

Average Function Value of Generation= 0.20637
Maximum Function Value = 0.99999

Number of Crossovers = 75
Elitist Reproduction on Individual 2

################# Generation 182 #################

#	Binary Code	Param1	Param2	Fitness
1	100110110000011000100010010000	0.6056	0.0669	0.03395
2	000100010001011000100010010000	0.0667	0.0669	0.99999
3	001110010001011000100110010000	0.2230	0.0747	0.22881
4	101100010001011000100110010000	0.6918	0.0747	0.05821
5	001100010001011000100110010000	0.1917	0.0747	0.00471
Average Values:		0.3558	0.0716	0.26513

Average Function Value of Generation= 0.26513
Maximum Function Value = 0.99999

Number of Crossovers = 66
Elitist Reproduction on Individual 5

```
################# Generation  183  #################
  #      Binary Code                Param1  Param2  Fitness
  1 100100010001011000100010010000  0.5668  0.0669  0.00000
  2 101100010001011000100010010000  0.6918  0.0669  0.06108
  3 000100010001011000100110010000  0.0667  0.0747  0.95295
  4 100110010000011000100010010000  0.5978  0.0669  0.01479
  5 000100010001011000100010010000  0.0667  0.0669  0.99999

 Average Values:                    0.3980  0.0685  0.40576

 Average Function Value of Generation= 0.40576
 Maximum Function Value             = 0.99999


  Number of Crossovers       =   87


%%%%%%%  Restart micro-population at generation  183  %%%%%%%


################# Generation  184  #################
  #      Binary Code                Param1  Param2  Fitness
  1 000100010001011000100010010000  0.0667  0.0669  0.99999
  2 010011010010010011000001100011  0.3013  0.3780  0.00006
  3 011100011111111000010101100011  0.4453  0.0421  0.28544
  4 111010110011101101001110110000  0.9189  0.6538  0.00010
  5 110100100011010000110101000100  0.8211  0.1036  0.01351

 Average Values:                    0.5107  0.2489  0.25982

 Average Function Value of Generation= 0.25982
 Maximum Function Value             = 0.99999


  Number of Crossovers       =   75
  Elitist Reproduction on Individual    3

################# Generation  185  #################
  #      Binary Code                Param1  Param2  Fitness
  1 000100010001111000100011100011  0.0669  0.0694  0.99478
  2 000100010101111000100011010010  0.0678  0.0689  0.99587
  3 000100010001011000100010010000  0.0667  0.0669  0.99999
  4 000100011111111000010111010010  0.0703  0.0455  0.69099
  5 001100011111111000100011000000  0.1953  0.0684  0.00980

 Average Values:                    0.0934  0.0638  0.73829

 Average Function Value of Generation= 0.73829
 Maximum Function Value             = 0.99999


  Number of Crossovers       =   73
  Elitist Reproduction on Individual    3

################# Generation  186  #################
  #      Binary Code                Param1  Param2  Fitness
  1 000100010001111000100010110010  0.0669  0.0679  0.99906
  2 000100010001011000100011010000  0.0667  0.0688  0.99684
  3 000100010001011000100010010000  0.0667  0.0669  0.99999
  4 000100010101111000100010010010  0.0678  0.0670  0.99920
  5 000100010001111000100011010010  0.0669  0.0689  0.99666
```

```
Average Values:                      0.0670  0.0679  0.99835

Average Function Value of Generation= 0.99835
Maximum Function Value              = 0.99999


 Number of Crossovers        =   76


################  Generation  187  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001111000100010010010    0.0669  0.0670  0.99999
  2 000100010001111000100010010010    0.0669  0.0670  0.99999
  3 000100010001111000100010010010    0.0669  0.0670  0.99999
  4 000100010001011000100010010000    0.0667  0.0669  0.99999
  5 000100010001111000100010010000    0.0669  0.0669  1.00000

Average Values:                       0.0668  0.0669  0.99999

Average Function Value of Generation= 0.99999
Maximum Function Value              = 1.00000


 Number of Crossovers        =   66

%%%%%%%  Restart micro-population at generation  187  %%%%%%%

################  Generation  188  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001111000100010010000    0.0669  0.0669  1.00000
  2 100111100001010000011000011100    0.6175  0.0477  0.06327
  3 001100010011100001001001010101    0.1923  0.1432  0.00001
  4 111000011010010001111000110100    0.8814  0.2360  0.01339
  5 100000111101111101010000110100    0.5151  0.6579  0.00531

Average Values:                       0.4546  0.2303  0.21639

Average Function Value of Generation= 0.21639
Maximum Function Value              = 1.00000


 Number of Crossovers        =   74
 Elitist Reproduction on Individual    5


################  Generation  189  ################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100111101111000110000110000    0.0776  0.0952  0.47851
  2 000101000001010000111010011100    0.0784  0.1141  0.13084
  3 100110010001110000101010010100    0.5981  0.0826  0.01266
  4 100110111001011101010000110100    0.6078  0.6579  0.00904
  5 000100010001111000100010010000    0.0669  0.0669  1.00000

Average Values:                       0.2858  0.2034  0.32621

Average Function Value of Generation= 0.32621
Maximum Function Value              = 1.00000


 Number of Crossovers        =   91
 Elitist Reproduction on Individual    2
```

```
#################  Generation  190  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000101010001110000111010011000  0.0825  0.1140  0.12136
  2 000100010001111000100010010000  0.0669  0.0669  1.00000
  3 000110010001110000101010010100  0.0981  0.0826  0.37307
  4 000100010001111000110000110000  0.0669  0.0952  0.52378
  5 000100010001111000110010110000  0.0669  0.0991  0.42903

 Average Values:                      0.0762  0.0916  0.48945

 Average Function Value of Generation= 0.48945
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   57
  Elitist Reproduction on Individual    4


#################  Generation  191  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001111000110010110000  0.0669  0.0991  0.42903
  2 000110010001110000101010010000  0.0981  0.0825  0.37420
  3 000100010001111000110010110000  0.0669  0.0991  0.42903
  4 000100010001111000100010010000  0.0669  0.0669  1.00000
  5 000100010001111000110000110000  0.0669  0.0952  0.52378

 Average Values:                      0.0731  0.0886  0.55121

 Average Function Value of Generation= 0.55121
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   67


#################  Generation  192  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001111000110010010000  0.0669  0.0981  0.45226
  2 000100010001111000110000010000  0.0669  0.0942  0.54802
  3 000100010001111000100010010000  0.0669  0.0669  1.00000
  4 000100010001111000110010110000  0.0669  0.0991  0.42903
  5 000100010001111000110010110000  0.0669  0.0991  0.42903

 Average Values:                      0.0669  0.0915  0.57167

 Average Function Value of Generation= 0.57167
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   76


%%%%%%%  Restart micro-population at generation  192  %%%%%%%


#################  Generation  193  #################
  #       Binary Code                 Param1  Param2  Fitness
  1 000100010001111000100010010000  0.0669  0.0669  1.00000
  2 111111110011001000011000101011  0.9969  0.0482  0.00197
  3 111111111011010100001010001101  0.9989  0.5199  0.00004
  4 110001010110101000010111100111  0.7712  0.0461  0.00004
  5 001101011110011000001010111011  0.2105  0.0213  0.01389
```

```
 Average Values:                       0.6089  0.1405  0.20319

 Average Function Value of Generation= 0.20319
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   66
  Elitist Reproduction on Individual    2


################  Generation  194  ################
  #      Binary Code                   Param1  Param2  Fitness
  1 010001010111101000100111100010  0.2714  0.0772  0.72539
  2 000100010001111000100010010000  0.0669  0.0669  1.00000
  3 101001011110101000010111101011  0.6481  0.0462  0.15940
  4 000100110011001000101000101001  0.0750  0.0794  0.83999
  5 001111110011101000010010100001  0.2470  0.0362  0.33361

 Average Values:                       0.2617  0.0612  0.61168

 Average Function Value of Generation= 0.61168
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   81
  Elitist Reproduction on Individual    1


################  Generation  195  ################
  #      Binary Code                   Param1  Param2  Fitness
  1 000100010001111000100010010000  0.0669  0.0669  1.00000
  2 000100010001011000101010101000  0.0667  0.0833  0.80951
  3 000101010011101000100110000010  0.0829  0.0743  0.78211
  4 000100010011011000101000000001  0.0672  0.0782  0.90482
  5 001110110001111000010010110000  0.2309  0.0366  0.18684

 Average Values:                       0.1029  0.0678  0.73665

 Average Function Value of Generation= 0.73665
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   61
  Elitist Reproduction on Individual    3


################  Generation  196  ################
  #      Binary Code                   Param1  Param2  Fitness
  1 000100010011111000100000010000  0.0674  0.0630  0.98842
  2 000100010001011000100010110000  0.0667  0.0679  0.99915
  3 000100010001111000100010010000  0.0669  0.0669  1.00000
  4 000100010001111000101010010000  0.0669  0.0825  0.82473
  5 000100010011011000101000000001  0.0672  0.0782  0.90482

 Average Values:                       0.0670  0.0717  0.94342

 Average Function Value of Generation= 0.94342
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   82
```

```
%%%%%%%  Restart micro-population at generation  196  %%%%%%%


#################  Generation  197  #################
  #        Binary Code                  Param1   Param2   Fitness
  1 000100010001111000100010010000  0.0669   0.0669   1.00000
  2 100111100101110111011110100000  0.6186   0.9346   0.00000
  3 101100011111000111100101010010  0.6951   0.9478   0.00000
  4 011001000011111001111110000111  0.3916   0.2463   0.00487
  5 001001111010001010110110111000  0.1548   0.3572   0.00000

 Average Values:                        0.3854   0.5106   0.20097

 Average Function Value of Generation= 0.20097
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   77
  Elitist Reproduction on Individual    5


#################  Generation  198  #################
  #        Binary Code                  Param1   Param2   Fitness
  1 011101010001111000101110010001  0.4575   0.0904   0.33195
  2 111001000011110101111110100111  0.8916   0.7473   0.00000
  3 010100010001111001100110010010  0.3169   0.1998   0.00118
  4 100101110001111111100010100000  0.5903   0.9424   0.00000
  5 000100010001111000100010010000  0.0669   0.0669   1.00000

 Average Values:                        0.4646   0.4094   0.26662

 Average Function Value of Generation= 0.26662
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   79
  Elitist Reproduction on Individual    5

#################  Generation  199  #################
  #        Binary Code                  Param1   Param2   Fitness
  1 010101010001111000100010010001  0.3325   0.0669   0.00536
  2 001101010001111000100010010000  0.2075   0.0669   0.05796
  3 010100010001111000100010010010  0.3169   0.0670   0.05695
  4 011101010001111000101010010000  0.4575   0.0825   0.42484
  5 000100010001111000100010010000  0.0669   0.0669   1.00000

 Average Values:                        0.2762   0.0700   0.30902

 Average Function Value of Generation= 0.30902
 Maximum Function Value              = 1.00000


  Number of Crossovers       =   74
  Elitist Reproduction on Individual    3

#################  Generation  200  #################
  #        Binary Code                  Param1   Param2   Fitness
  1 000101010001111000100010010000  0.0825   0.0669   0.82536
  2 000100010001111000101010010000  0.0669   0.0825   0.82473
  3 000100010001111000100010010000  0.0669   0.0669   1.00000
  4 001100010001111000100010010000  0.1919   0.0669   0.00507
  5 010101010001111000101010010000  0.3325   0.0825   0.00442
```

Average Values: 0.1481 0.0731 0.53192

Average Function Value of Generation= 0.53192
Maximum Function Value = 1.00000

Number of Crossovers = 85

%%%%%%% Restart micro-population at generation 200 %%%%%%%

Summary of Output

Generation	Evaluations	Avg.Fitness	Best Fitness
0.1000E+01	0.5000E+01	0.2047E-01	0.10147E+00
0.2000E+01	0.1000E+02	0.4206E-01	0.10147E+00
0.3000E+01	0.1500E+02	0.1302E+00	0.22471E+00
0.4000E+01	0.2000E+02	0.4967E-01	0.22471E+00
0.5000E+01	0.2500E+02	0.4622E-01	0.22471E+00
0.6000E+01	0.3000E+02	0.6616E-01	0.22471E+00
0.7000E+01	0.3500E+02	0.6470E-01	0.22471E+00
0.8000E+01	0.4000E+02	0.6744E-01	0.22471E+00
0.9000E+01	0.4500E+02	0.6783E-01	0.22471E+00
0.1000E+02	0.5000E+02	0.4531E-01	0.22471E+00
0.1100E+02	0.5500E+02	0.5091E-01	0.22471E+00
0.1200E+02	0.6000E+02	0.9019E-01	0.22492E+00
0.1300E+02	0.6500E+02	0.9066E-01	0.22492E+00
0.1400E+02	0.7000E+02	0.9173E-01	0.22492E+00
0.1500E+02	0.7500E+02	0.1355E+00	0.29078E+00
0.1600E+02	0.8000E+02	0.1002E+00	0.29078E+00
0.1700E+02	0.8500E+02	0.2308E+00	0.57101E+00
0.1800E+02	0.9000E+02	0.4515E+00	0.57101E+00
0.1900E+02	0.9500E+02	0.1247E+00	0.57101E+00
0.2000E+02	0.1000E+03	0.1201E+00	0.57101E+00
0.2100E+02	0.1050E+03	0.1733E+00	0.57101E+00
0.2200E+02	0.1100E+03	0.2699E+00	0.57101E+00
0.2300E+02	0.1150E+03	0.3930E+00	0.57101E+00
0.2400E+02	0.1200E+03	0.1192E+00	0.57101E+00
0.2500E+02	0.1250E+03	0.2650E+00	0.57101E+00
0.2600E+02	0.1300E+03	0.4157E+00	0.57101E+00
0.2700E+02	0.1350E+03	0.4508E+00	0.57370E+00
0.2800E+02	0.1400E+03	0.1211E+00	0.57370E+00
0.2900E+02	0.1450E+03	0.1256E+00	0.57370E+00
0.3000E+02	0.1500E+03	0.1260E+00	0.57370E+00
0.3100E+02	0.1550E+03	0.1215E+00	0.57370E+00
0.3200E+02	0.1600E+03	0.3631E+00	0.65745E+00
0.3300E+02	0.1650E+03	0.3706E+00	0.65745E+00
0.3400E+02	0.1700E+03	0.1362E+00	0.65745E+00
0.3500E+02	0.1750E+03	0.1739E+00	0.65745E+00
0.3600E+02	0.1800E+03	0.2188E+00	0.65745E+00
0.3700E+02	0.1850E+03	0.4179E+00	0.66297E+00
0.3800E+02	0.1900E+03	0.5469E+00	0.66297E+00
0.3900E+02	0.1950E+03	0.6606E+00	0.66672E+00
0.4000E+02	0.2000E+03	0.6645E+00	0.66797E+00
0.4100E+02	0.2050E+03	0.1348E+00	0.66797E+00
0.4200E+02	0.2100E+03	0.1927E+00	0.66797E+00
0.4300E+02	0.2150E+03	0.2034E+00	0.66797E+00
0.4400E+02	0.2200E+03	0.3022E+00	0.66797E+00
0.4500E+02	0.2250E+03	0.4311E+00	0.67412E+00
0.4600E+02	0.2300E+03	0.1356E+00	0.67412E+00
0.4700E+02	0.2350E+03	0.1629E+00	0.67412E+00
0.4800E+02	0.2400E+03	0.1847E+00	0.67412E+00
0.4900E+02	0.2450E+03	0.1774E+00	0.67412E+00

0.5000E+02	0.2500E+03	0.6680E+00	0.98354E+00
0.5100E+02	0.2550E+03	0.7734E+00	0.98354E+00
0.5200E+02	0.2600E+03	0.1970E+00	0.98354E+00
0.5300E+02	0.2650E+03	0.2277E+00	0.98354E+00
0.5400E+02	0.2700E+03	0.1970E+00	0.98354E+00
0.5500E+02	0.2750E+03	0.3008E+00	0.98354E+00
0.5600E+02	0.2800E+03	0.3150E+00	0.98354E+00
0.5700E+02	0.2850E+03	0.3750E+00	0.98354E+00
0.5800E+02	0.2900E+03	0.2041E+00	0.98354E+00
0.5900E+02	0.2950E+03	0.3842E+00	0.98354E+00
0.6000E+02	0.3000E+03	0.3195E+00	0.98354E+00
0.6100E+02	0.3050E+03	0.5175E+00	0.98414E+00
0.6200E+02	0.3100E+03	0.6032E+00	0.99008E+00
0.6300E+02	0.3150E+03	0.1982E+00	0.99008E+00
0.6400E+02	0.3200E+03	0.2733E+00	0.99008E+00
0.6500E+02	0.3250E+03	0.3861E+00	0.99008E+00
0.6600E+02	0.3300E+03	0.6052E+00	0.99929E+00
0.6700E+02	0.3350E+03	0.2485E+00	0.99929E+00
0.6800E+02	0.3400E+03	0.3533E+00	0.99929E+00
0.6900E+02	0.3450E+03	0.2072E+00	0.99929E+00
0.7000E+02	0.3500E+03	0.2748E+00	0.99929E+00
0.7100E+02	0.3550E+03	0.4527E+00	0.99929E+00
0.7200E+02	0.3600E+03	0.6767E+00	0.99929E+00
0.7300E+02	0.3650E+03	0.7503E+00	0.99929E+00
0.7400E+02	0.3700E+03	0.9653E+00	0.99929E+00
0.7500E+02	0.3750E+03	0.9653E+00	0.99929E+00
0.7600E+02	0.3800E+03	0.2080E+00	0.99929E+00
0.7700E+02	0.3850E+03	0.3159E+00	0.99929E+00
0.7800E+02	0.3900E+03	0.4895E+00	0.99929E+00
0.7900E+02	0.3950E+03	0.9026E+00	0.99936E+00
0.8000E+02	0.4000E+03	0.9606E+00	0.99940E+00
0.8100E+02	0.4050E+03	0.2003E+00	0.99940E+00
0.8200E+02	0.4100E+03	0.2131E+00	0.99940E+00
0.8300E+02	0.4150E+03	0.2240E+00	0.99940E+00
0.8400E+02	0.4200E+03	0.4382E+00	0.99940E+00
0.8500E+02	0.4250E+03	0.7995E+00	0.99940E+00
0.8600E+02	0.4300E+03	0.2000E+00	0.99940E+00
0.8700E+02	0.4350E+03	0.2002E+00	0.99940E+00
0.8800E+02	0.4400E+03	0.2034E+00	0.99940E+00
0.8900E+02	0.4450E+03	0.2150E+00	0.99940E+00
0.9000E+02	0.4500E+03	0.2403E+00	0.99940E+00
0.9100E+02	0.4550E+03	0.2754E+00	0.99940E+00
0.9200E+02	0.4600E+03	0.3286E+00	0.99940E+00
0.9300E+02	0.4650E+03	0.6486E+00	0.99948E+00
0.9400E+02	0.4700E+03	0.5697E+00	0.99948E+00
0.9500E+02	0.4750E+03	0.2615E+00	0.99948E+00
0.9600E+02	0.4800E+03	0.2865E+00	0.99948E+00
0.9700E+02	0.4850E+03	0.4980E+00	0.99967E+00
0.9800E+02	0.4900E+03	0.8079E+00	0.99967E+00
0.9900E+02	0.4950E+03	0.2000E+00	0.99967E+00
0.1000E+03	0.5000E+03	0.2000E+00	0.99967E+00
0.1010E+03	0.5050E+03	0.2233E+00	0.99967E+00
0.1020E+03	0.5100E+03	0.5961E+00	0.99967E+00
0.1030E+03	0.5150E+03	0.7979E+00	0.99967E+00
0.1040E+03	0.5200E+03	0.2023E+00	0.99967E+00
0.1050E+03	0.5250E+03	0.2029E+00	0.99967E+00
0.1060E+03	0.5300E+03	0.4054E+00	0.99967E+00
0.1070E+03	0.5350E+03	0.4018E+00	0.99967E+00
0.1080E+03	0.5400E+03	0.2025E+00	0.99967E+00
0.1090E+03	0.5450E+03	0.2005E+00	0.99967E+00
0.1100E+03	0.5500E+03	0.4374E+00	0.99967E+00
0.1110E+03	0.5550E+03	0.4084E+00	0.99967E+00
0.1120E+03	0.5600E+03	0.8017E+00	0.99976E+00
0.1130E+03	0.5650E+03	0.9978E+00	0.99976E+00

0.1140E+03	0.5700E+03	0.2023E+00	0.99976E+00
0.1150E+03	0.5750E+03	0.2070E+00	0.99976E+00
0.1160E+03	0.5800E+03	0.2815E+00	0.99976E+00
0.1170E+03	0.5850E+03	0.7606E+00	0.99976E+00
0.1180E+03	0.5900E+03	0.9605E+00	0.99976E+00
0.1190E+03	0.5950E+03	0.2024E+00	0.99976E+00
0.1200E+03	0.6000E+03	0.2120E+00	0.99976E+00
0.1210E+03	0.6050E+03	0.2099E+00	0.99976E+00
0.1220E+03	0.6100E+03	0.4567E+00	0.99976E+00
0.1230E+03	0.6150E+03	0.8154E+00	0.99976E+00
0.1240E+03	0.6200E+03	0.2712E+00	0.99976E+00
0.1250E+03	0.6250E+03	0.3365E+00	0.99976E+00
0.1260E+03	0.6300E+03	0.9095E+00	0.99976E+00
0.1270E+03	0.6350E+03	0.9518E+00	0.99976E+00
0.1280E+03	0.6400E+03	0.2006E+00	0.99976E+00
0.1290E+03	0.6450E+03	0.4257E+00	0.99976E+00
0.1300E+03	0.6500E+03	0.3505E+00	0.99976E+00
0.1310E+03	0.6550E+03	0.7397E+00	0.99999E+00
0.1320E+03	0.6600E+03	0.8924E+00	0.99999E+00
0.1330E+03	0.6650E+03	0.9999E+00	0.99999E+00
0.1340E+03	0.6700E+03	0.2462E+00	0.99999E+00
0.1350E+03	0.6750E+03	0.3022E+00	0.99999E+00
0.1360E+03	0.6800E+03	0.5006E+00	0.99999E+00
0.1370E+03	0.6850E+03	0.9999E+00	0.99999E+00
0.1380E+03	0.6900E+03	0.2333E+00	0.99999E+00
0.1390E+03	0.6950E+03	0.2429E+00	0.99999E+00
0.1400E+03	0.7000E+03	0.4042E+00	0.99999E+00
0.1410E+03	0.7050E+03	0.8887E+00	0.99999E+00
0.1420E+03	0.7100E+03	0.5822E+00	0.99999E+00
0.1430E+03	0.7150E+03	0.9595E+00	0.99999E+00
0.1440E+03	0.7200E+03	0.2022E+00	0.99999E+00
0.1450E+03	0.7250E+03	0.3399E+00	0.99999E+00
0.1460E+03	0.7300E+03	0.5479E+00	0.99999E+00
0.1470E+03	0.7350E+03	0.4937E+00	0.99999E+00
0.1480E+03	0.7400E+03	0.8568E+00	0.99999E+00
0.1490E+03	0.7450E+03	0.2559E+00	0.99999E+00
0.1500E+03	0.7500E+03	0.3557E+00	0.99999E+00
0.1510E+03	0.7550E+03	0.5400E+00	0.99999E+00
0.1520E+03	0.7600E+03	0.6019E+00	0.99999E+00
0.1530E+03	0.7650E+03	0.2112E+00	0.99999E+00
0.1540E+03	0.7700E+03	0.2854E+00	0.99999E+00
0.1550E+03	0.7750E+03	0.4459E+00	0.99999E+00
0.1560E+03	0.7800E+03	0.3944E+00	0.99999E+00
0.1570E+03	0.7850E+03	0.3577E+00	0.99999E+00
0.1580E+03	0.7900E+03	0.4808E+00	0.99999E+00
0.1590E+03	0.7950E+03	0.2023E+00	0.99999E+00
0.1600E+03	0.8000E+03	0.3414E+00	0.99999E+00
0.1610E+03	0.8050E+03	0.3607E+00	0.99999E+00
0.1620E+03	0.8100E+03	0.5577E+00	0.99999E+00
0.1630E+03	0.8150E+03	0.7863E+00	0.99999E+00
0.1640E+03	0.8200E+03	0.2000E+00	0.99999E+00
0.1650E+03	0.8250E+03	0.2209E+00	0.99999E+00
0.1660E+03	0.8300E+03	0.4072E+00	0.99999E+00
0.1670E+03	0.8350E+03	0.7932E+00	0.99999E+00
0.1680E+03	0.8400E+03	0.3731E+00	0.99999E+00
0.1690E+03	0.8450E+03	0.4897E+00	0.99999E+00
0.1700E+03	0.8500E+03	0.3578E+00	0.99999E+00
0.1710E+03	0.8550E+03	0.3052E+00	0.99999E+00
0.1720E+03	0.8600E+03	0.6876E+00	0.99999E+00
0.1730E+03	0.8650E+03	0.9976E+00	0.99999E+00
0.1740E+03	0.8700E+03	0.9975E+00	0.99999E+00
0.1750E+03	0.8750E+03	0.2349E+00	0.99999E+00
0.1760E+03	0.8800E+03	0.3260E+00	0.99999E+00
0.1770E+03	0.8850E+03	0.3816E+00	0.99999E+00

0.1780E+03	0.8900E+03	0.7423E+00	0.99999E+00
0.1790E+03	0.8950E+03	0.2007E+00	0.99999E+00
0.1800E+03	0.9000E+03	0.2175E+00	0.99999E+00
0.1810E+03	0.9050E+03	0.2064E+00	0.99999E+00
0.1820E+03	0.9100E+03	0.2651E+00	0.99999E+00
0.1830E+03	0.9150E+03	0.4058E+00	0.99999E+00
0.1840E+03	0.9200E+03	0.2598E+00	0.99999E+00
0.1850E+03	0.9250E+03	0.7383E+00	0.99999E+00
0.1860E+03	0.9300E+03	0.9983E+00	0.99999E+00
0.1870E+03	0.9350E+03	0.1000E+01	0.10000E+01
0.1880E+03	0.9400E+03	0.2164E+00	0.10000E+01
0.1890E+03	0.9450E+03	0.3262E+00	0.10000E+01
0.1900E+03	0.9500E+03	0.4894E+00	0.10000E+01
0.1910E+03	0.9550E+03	0.5512E+00	0.10000E+01
0.1920E+03	0.9600E+03	0.5717E+00	0.10000E+01
0.1930E+03	0.9650E+03	0.2032E+00	0.10000E+01
0.1940E+03	0.9700E+03	0.6117E+00	0.10000E+01
0.1950E+03	0.9750E+03	0.7367E+00	0.10000E+01
0.1960E+03	0.9800E+03	0.9434E+00	0.10000E+01
0.1970E+03	0.9850E+03	0.2010E+00	0.10000E+01
0.1980E+03	0.9900E+03	0.2666E+00	0.10000E+01
0.1990E+03	0.9950E+03	0.3090E+00	0.10000E+01
0.2000E+03	0.1000E+04	0.5319E+00	0.10000E+01

201 5

1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0
2	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0
3	0	1	1	0	1	1	0	0	1	1	1	0	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	1	0
4	0	1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	0	1	1	0	1	0	0	0	1	0	0	1	0	1
5	0	0	1	1	0	1	1	0	1	0	0	1	1	1	0	1	0	0	1	1	0	0	0	0	0	1	0	0	1	0

```
      parameter (indmax=200,nchrmax=30,nparmax=2)
c  indmax  = maximum # of individuals, i.e. max population size
c  nchrmax = maximum # of chromosomes (binary bits) per individual
c  nparmax = maximum # of parameters which the chromosomes make up
```

D.L. Carroll's FORTRAN Genetic Algorithm Driver

This is version 1.7, last updated on 12/11/98.
Download from: <http://www.staff.uiuc.edu/~carroll/ga.html>

This genetic algorithm (GA) driver is free for public use. My only request is that the user reference and/or acknowledge the use of this driver in any papers/reports/articles which have results obtained from the use of this driver. I would also appreciate a copy of such papers/articles/reports, or at least an e-mail message with the reference so I can get a copy. Thanks.

This program is a FORTRAN version of a genetic algorithm driver. This code initializes a random sample of individuals with different parameters to be optimized using the genetic algorithm approach, i.e. evolution via survival of the fittest. The selection scheme used is tournament selection with a shuffling technique for choosing random pairs for mating. The routine includes binary coding for the individuals, jump mutation, creep mutation, and the option for single-point or uniform crossover. Niching (sharing) and an option for the number of children per pair of parents has been added. More recently, an option for the use of a micro-GA has been added.

For companies wishing to link this GA driver with an existing code, I am available for some consulting work. Regardless, I suggest altering this code as little as possible to make future updates easier to incorporate.

Any users new to the GA world are encouraged to read David Goldberg's "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

```
The seven FORTRAN GA files are:   ga170.f
                                  ga.inp
                                  ga2.inp (w/ different namelist identifier)
                                  ga.out
                                  ga.restart
                                  params.f
                                  ReadMe (this file!)
```

I have provided a sample subroutine "func", but ultimately the user must supply this subroutine "func" which should be your cost function. You should be able to run the code with the sample subroutine "func" and the provided ga.inp file and obtain the optimal function value of 1.0000 at generation 187 with the uniform crossover micro-GA enabled (this is 935 function evaluations). Note that because different computers may treat precision and truncation differently, I have seen cases where two computers using the same input produce different evolution histories (but still converge to the optimal).

I still recommend using the micro-GA technique (microga=1) with uniform crossover (iunifrm=1). However, if possible, I strongly suggest that you use values of nposibl of 2**n (2, 4, 8, 16, 32, 64, etc.). While my test function works fine for other values of nposibl, I have encountered problems where the uniform crossover micro-GA has difficulty with parameters having long bit strings and a non-2**n value of nposibl, e.g. nposibl=1000, will have 10 bits assigned (for this case

I would suggest running nposibl=1024 rather than 1000); I am presently investigating possible fixes for this situation.

Updates:

Version 1.7 includes several improvements:

(i) The coding and input files are cleaned up to provide identical output across a wider range of computers.

(ii) The arrays have been rearranged to enable a more efficient caching of system memory. For cases with very large population sizes, run time improvements of as much as a factor of 4-6 were observed! For population sizes less than 1000 you will not see much change.

(iii) A summary of the results has been added to the end of the output file.

(iv) An alternate input file "ga2.inp" has been included. Some compilers require an '&' and a '/' in the namelist input file, rather than '$' signs.

(v) For those wishing to try ever harder test functions, the included function is now N-dimensional, where N is simply determined by the number of parameters specified (nparam).

Version 1.6.5 of the code allowed creep mutations to be implemented with the micro-GA technique. (This version was never officially released.)

Version 1.6.4 of the code has a minor modification to the niching routine and another minor modification which would only affect a user having a single parameter with more than 2**30 possibilities (probably noone has used this large a number).

Version 1.6.3 of the code fixes a bug in the niching routine. Niching should now work much better than in previous versions. A few other minor changes have been made (not worth mentioning). The sample function has been changed to something a bit more challenging.

Version 1.6.2 of the code has had major restructuring in the form of converting all of the operators (crossover, mutation, etc.) into subroutines. The code logic should be a little more understandable now and it lends itself to more easily modifying parts of the code. The counter kountmx (see v1.6.1 comments below) was added to the namelist input. Otherwise, code performance should be the same.

Version 1.6.1 of the code has very minor modifications. If you are already successfully using the code, then you will not need this update.

(i) Added a little documentation about changing format statements 1050, 1075, 1275, and 1500 when you change nparam or the total number of chromosomes (see below).

(ii) I have commented out all of the lines of code dealing with cputime. The Macintosh specific SECNDS call was causing more questions than I had anticipated. However, other than commenting the lines out, I have left them in their location for reference in case the user wants a cputime added.

(iii) I have included a sample output file.

(iv) Added counter (kountmx) to control how frequently the restart file is written. This saves I/O time and wear and tear on storage device. Presently set to write every fifth generation.

Version 1.6 of the code has incorporated the ability to use a micro-GA approach; this significantly reduced the number of function evaluations to find the global maximum of my test function.

Version 1.5 of the code has added some more flexibility to your

available options:

(i) You now specify the minimum and maximum values of the parameters rather than the minimum and the increment.

(ii) You now specify the number of possibilities you want for each parameter, not the number of bits. This modification has two features: first, the program automatically calculates the number of bits per parameter; second, you are no longer forced to have a number of possibilities equal to 2**n. While the code is more efficient when there 2**n possibilities per parameter, it will run quite well with a lesser number; e.g. a colleague has 25 specific airfoil families he wants to investigate, greater than 16, less than 32.

(iii) You can now specify specific parameters for niching. Earlier versions of the code forced you to niche on all parameters. Now, the input array 'nichflg' permits you to choose the parameters for niching.

(iv) You have an input flag to prevent the printing of specific jump. and creep mutation information

(v) You now specify the maximum values of population size, number of parameters and number of chromosomes in an include file (params.f). This sets the maximum array sizes in the code. When running, the code only uses the array size up to npopsiz and nparam (from ga.inp) and nchrome (computed internally from the nposibl input array).

The code is presently set for a maximum population size of 200, 30 chromosomes (binary bits) and 2 parameters. These values can be changed in params.f as appropriate for your problem. Correspondingly you will have to change a few 'write' and 'format' statements if you change nchrmax and/or nparmax. In particular, if you change nchrome and/or nparam, then you should change the 'format' statement numbers 1050, 1075, 1275, and 1500. For example, if you have a problem with 4 parameters and 16 chromosomes (bits), then you should change these format statements to be:

```
 1050 format(1x,' #      Binary Code',8x,'Param1 Param2 Param3',
     +            ' Param4 Fitness')
 1075 format(i3,1x,16i1,4(1x,f6.2),1x,f6.2)
 1275 format(/' Average Values:',10x,4(1x,f6.2),1x,f6.2/)
 1500 format(i5,3x,16i2)
```

The CPU time related lines of code reference a Macintosh specific time function (SECNDS). To avoid compiler errors with other computers, I have commented out these lines of code. If you wish to have cputime output,then you will have to change the time functions for the specific computer you are running on. Most modern Unix machines will recognize the 'etime' function; these lines are added to the code along with the variable 'tarray' and 'cpu...again, to avoid compiler errors with different computers, these lines of code are also commented out.

A common problem arises with the Microsoft PowerStation compiler, i.e., PowerStation does not recognize the abbreviation NML for NAMELIST. If you are using PowerStation, you will likely have to substitute NAMELIST for all instances of NML.

Please feel free to contact me with questions, comments, or errors (hopefully none of latter).

Enjoy!

David L. Carroll
University of Illinois
140 Mechanical Engineering Bldg.
1206 W. Green Street
Urbana, IL 61801

e-mail: carroll@uiuc.edu
Phone: 217-333-4741
fax: 217-244-6534

##

micro-GA Tip:

My favorite GA technique is still the micro-GA. At this point, I recommend using the micro-GA with uniform crossover and a small population size. The following inputs gave me excellent performance:

```
microga = 1
npopsiz = 5
maxgen  = 100
iunifrm = 1
```

I have also gotten good performance with the single-point crossover (iunifrm=0), micro-GA.

If you decide to use the micro-GA, you will not need to worry about the population sizing or creep mutation tips below.

See the Krishnakumar reference below for more information about micro-GA's.

##

Population Sizing Tip:

I've had a lot of people ask me about population sizing, especially people who are attempting large problems where 100 individuals is probably not enough. The true authority on the subject is David Goldberg, but here is a crude population scaling law in my paper (based on Goldberg & Deb, 1992):

```
npopsiz = order[(l/k)(2**k)] for binary coding
```

where l = nchrome and k is the average size of the schema of interest (effectively the average number of bits per parameter, i.e. approximately equal to nchrome/nparam, rounded to the nearest integer). I find that when I have uniform crossover and niching turned on (which I recommend doing), that this scaling law is usually overkill, i.e. you can most likely get by with populations at least twice as small.

Remember to make the parameter 'indmax' (in 'params.f') greater than or equal to 'npopsiz'.

##

Creep Mutation Probability Tip:

I generally like to have approximately the same number of creep mutations and jump mutations per generation. Using basic probabilistic arguments, it can be shown that you will get approximately the same number of creep and jump mutations when

```
pcreep = (nchrome/nparam) * pmutate
```

where pmutate (the jump mutation probability) is 1/npopsiz.

##

Suggested reading that I have found to be of use:

Goldberg, D. E., and Richardson, J., "Genetic Algorithms with

Sharing for Multimodal Function Optimization," Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 41-49.

Goldberg, D. E., "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

Goldberg, D. E., "A Note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing," in: Complex Systems, Vol. 4, Complex Systems Publications, Inc., 1990, pp. 445-460.

Goldberg, D. E., "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking," in: Complex Systems, Vol. 5, Complex Systems Publications, Inc., 1991, pp. 139-167.

Goldberg, D. E., and Deb, K., "A Comparitive Analysis of Selection Schemes Used in Genetic Algorithms," in: Foundations of Genetic Algorithms, ed. by Rawlins, G.J.E., Morgan Kaufmann Publishers, San Mateo, CA, pp. 69-93, 1991.

Goldberg, D. E., Deb, K., and Clark, J. H., "Genetic Algorithms, Noise, and the Sizing of Populations," in: Complex Systems, Vol. 6, Complex Systems Pub., Inc., 1992, pp. 333-362.

Krishnakumar, K., "Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization," SPIE: Intelligent Control and Adaptive Systems, Vol. 1196, Philadelphia, PA, 1989.

Syswerda, G., "Uniform Crossover in Genetic Algorithms," in: Proceedings of the Third International Conference on Genetic Algorithms, Schaffer, J. (Ed.), Morgan Kaufmann Publishers, Los Altos, CA, pp. 2-9, 1989.

##

If you are interested in my work (which may give some insights into how and why I coded some aspects of my GA), I can mail copies of three papers of mine.

G. Yang, L.E. Reinstein, S. Pai, Z. Xu, and D.L. Carroll, "A new genetic algorithm technique in optimization of permanent 125-I prostate implants," Medical Physics, Vol. 25, No. 12, 1998, pp. 2308-2315.

Carroll, D. L., "Chemical Laser Modeling with Genetic Algorithms," AIAA J., Vol. 34, 2, 1996, pp.338-346.
(A preprint version of this paper can now be downloaded in PDF format via my website: <http://www.staff.uiuc.edu/~carroll/gatips.html> look for AIAA1996.pdf)

Carroll, D. L., "Genetic Algorithms and Optimizing Chemical Oxygen-Iodine Lasers," Developments in Theoretical and Applied Mechanics, Vol. XVIII, eds. H.B. Wilson, R.C. Batra, C.W. Bert, A.M.J. Davis, R.A. Schapery, D.S. Stewart, and F.F. Swinson, School of Engineering, The University of Alabama, 1996, pp.411-424.
(This paper can now be downloaded in PDF format via my website: <http://www.staff.uiuc.edu/~carroll/gatips.html> look for SECTAM18.pdf)

##

Disclaimer: this program is not guaranteed to be free of error (although it is believed to be free of error), therefore it should not be relied on for solving problems where an error could result in injury or loss. If this code is used for such solutions, it is

entirely at the user's risk and the author disclaims all liability.

The following portion of this disclosure was created in *Powerpoint* for purposes of further describing the present invention. It particularly concerns bandwidth enhanced normal mode helical antennas. It begins by setting forth the objectives, considerations, and questions addressed in the beginning stages of development of the present invention. The affects of different physical antenna parameters on antenna performance are addressed by showing the affect on the VSWR by these variations.

The remainder of the following disclosure portion shows several different antenna designs and in graphical form illustrates the respective performance of each. A straight wire antenna, a simple helix, and a triple helix are all examined. Each antenna is modified by the addition of various combinations of parasitic elements. The characteristics of each of these antennas are then illustrated. The VSWR, directivity, and input impedance are shown so that the different antennas having different combinations of parasitic elements can be analyzed effectively.

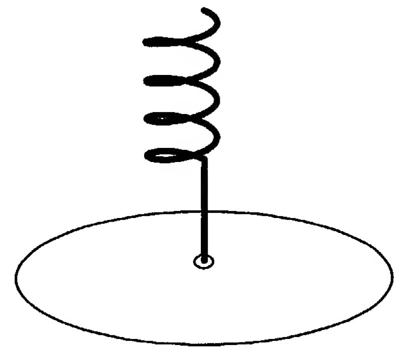
This portion of the disclosure concludes by summarizing the results obtained from the different combinations. The conclusions drawn from these results are then set forth. It illustrates the initial indications that bandwidth improvements could be made by the addition of these parasitic elements.

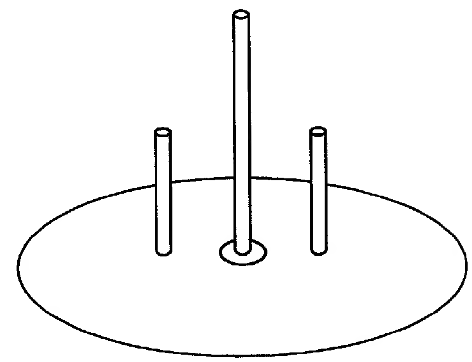
Overview

- Introduction
- Helix geometry considerations
- Procedure for efficient optimization of helix with parasitic elements
- Numerical results for open sleeve monopole
- Numerical results for bandwidth improvement of normal mode helix
- Conclusions

Introduction

Objectives for Antenna
- Low-profile
- Omnidirectional
- Broadband

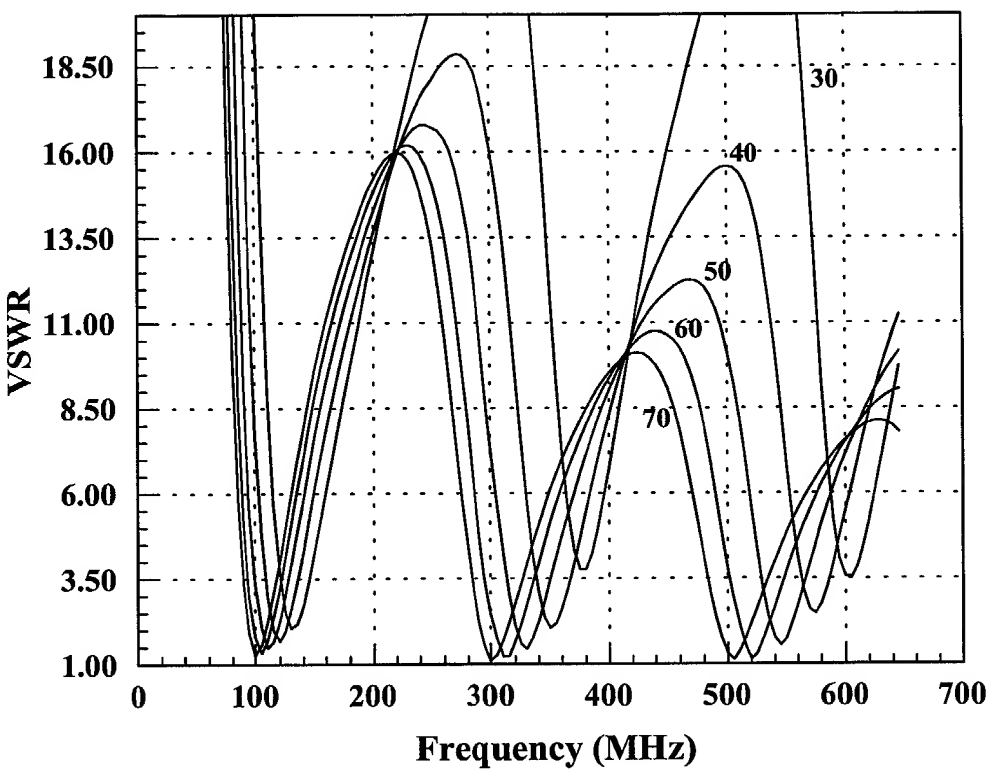




Considerations
- Helix can be made shorter by adjusting the pitch
- Normal mode helix has narrow bandwidth
- Parasitic elements increase the bandwidth of straight wires

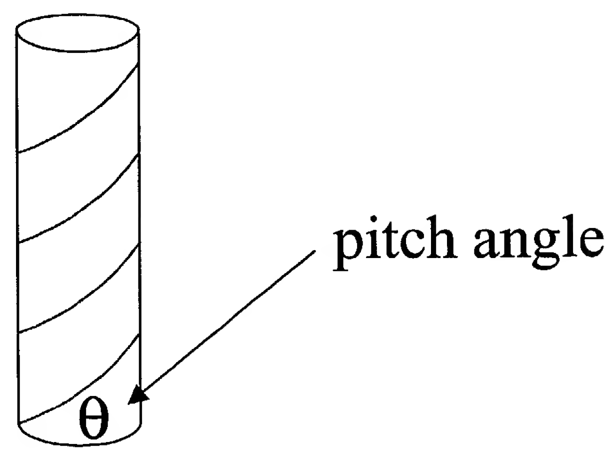
Questions addressed
- Can the bandwidth of the normal mode helix be improved with parasitic elements?
- If so, what are suitable structures for the parasites?

Effect of Pitch Angle on Helix VSWR



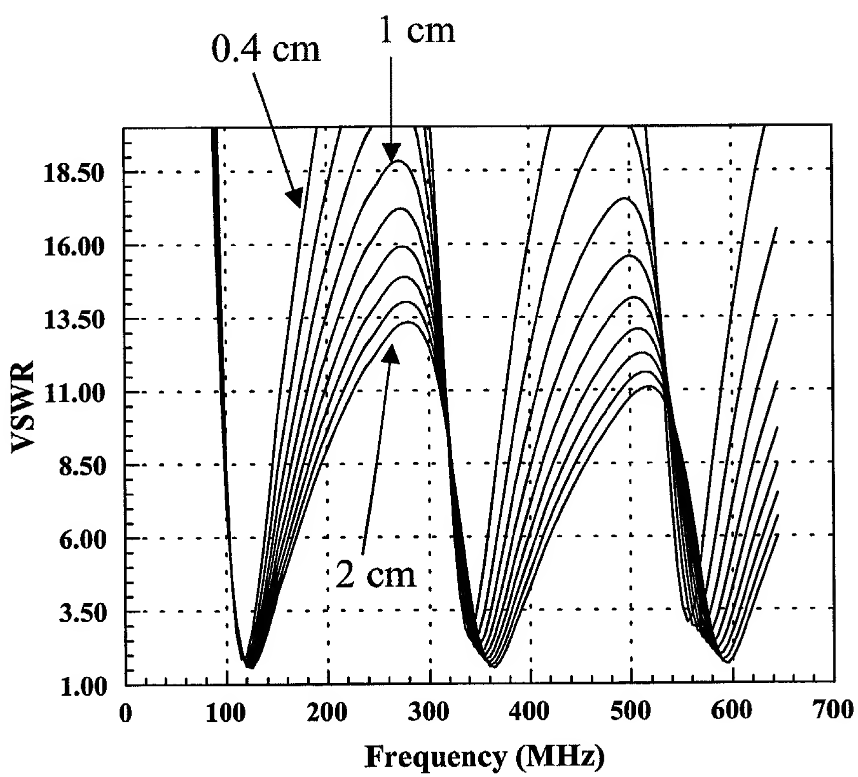
Pitch (Degrees)	Height Reduction (%)	Total Height (cm)
30	45.0	41.3
40	32.0	50.9
50	21.0	59.2
60	12.0	66.0
70	5.4	70.9

Total wire length 75 cm
wire radius 0.5 cm



Numerical results

Effect of Wire Radius on Helix VSWR

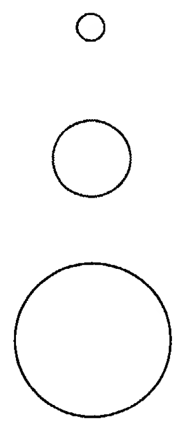


Numerical results

Helix Geometry

Height of straight wire	7.5 cm
Circumference of one turn	15 cm
Total wire length	75 cm
Number of turns	4.5
Pitch angle	40 degrees

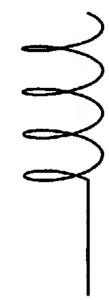
Diameter



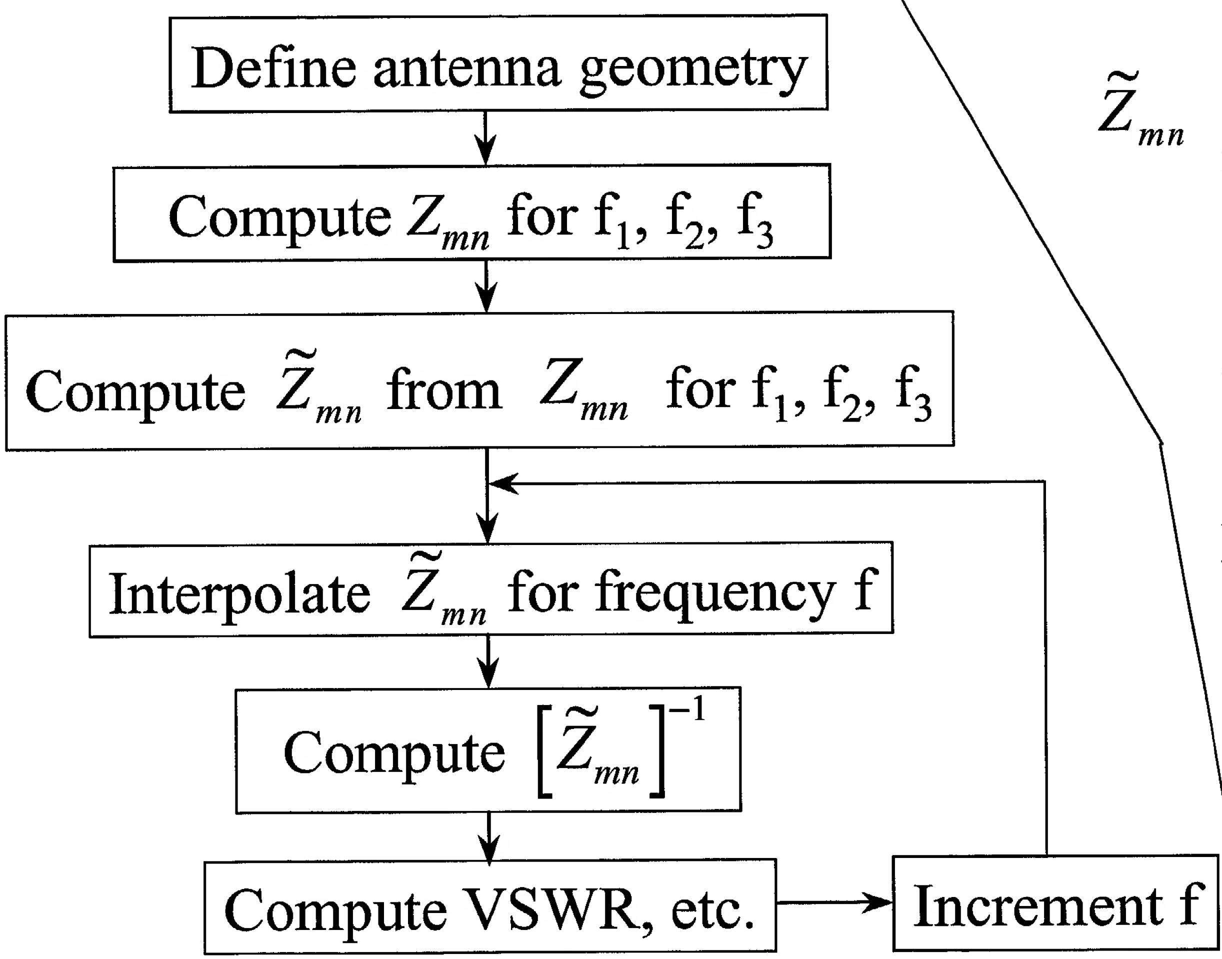
0.4 cm

1 cm

2 cm



Efficient Evaluation of Antennas



$\tilde{Z}_{mn}$ is the reduced-rank impedance matrix used for curved wires.

This is done for each antenna in the sample population.

References on Reduced-Rank Matrices

S.D. Rogers and C.M. Butler, “Reduced Rank Matrices for Curved Wire Structures,” Digest of 1997 Antennas and Propagation Society (APS) Symposium, Montreal, Canada, vol. 1, pp. 68-71, July 1997.

S.D. Rogers, “Efficient Numerical Techniques for Curved Wires,” Masters Thesis, Clemson University, August 1997.

Website for above liturature and copies of these slides:
www.eng.clemson.edu/~sdroger

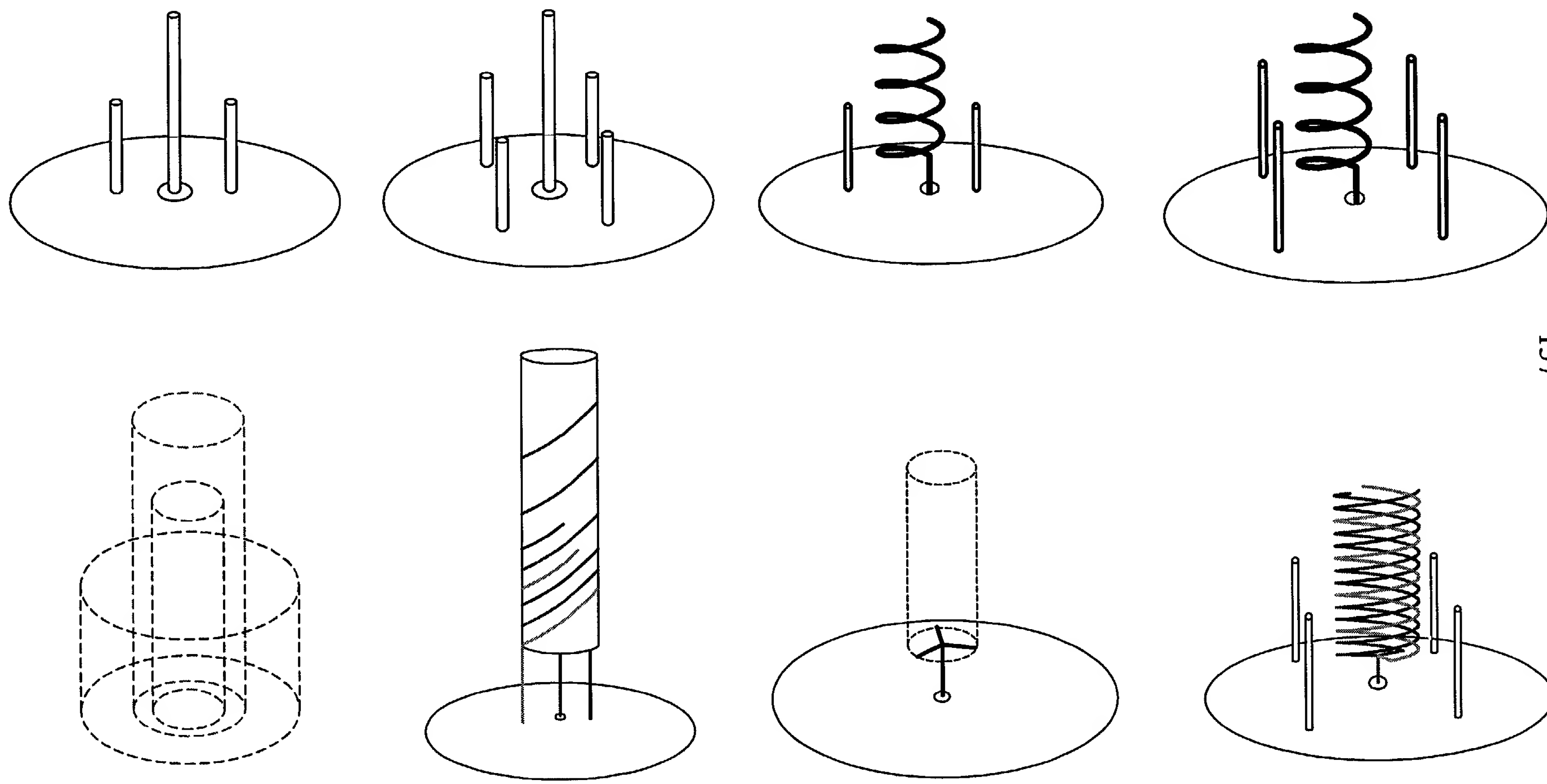
Reference

Genetic algorithm driver:

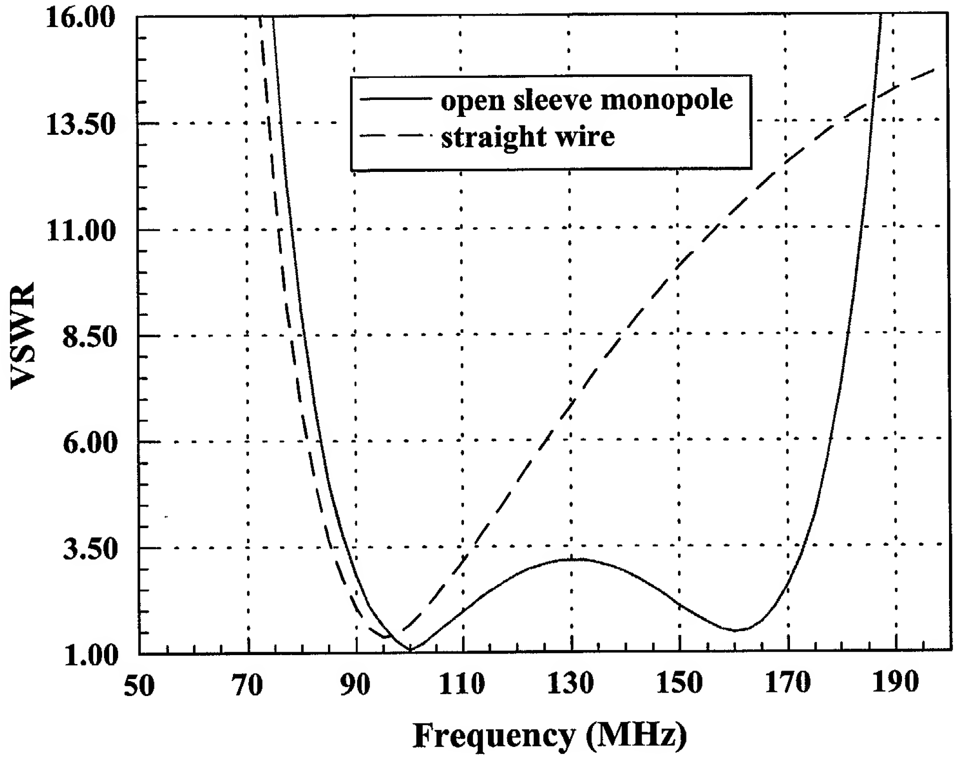
Carroll, D.L., "A FORTRAN Genetic Algorithm Code",
Univ. of Illinois, Urbana IL.
http://www.staff.uiuc.edu/~carroll/ga.html

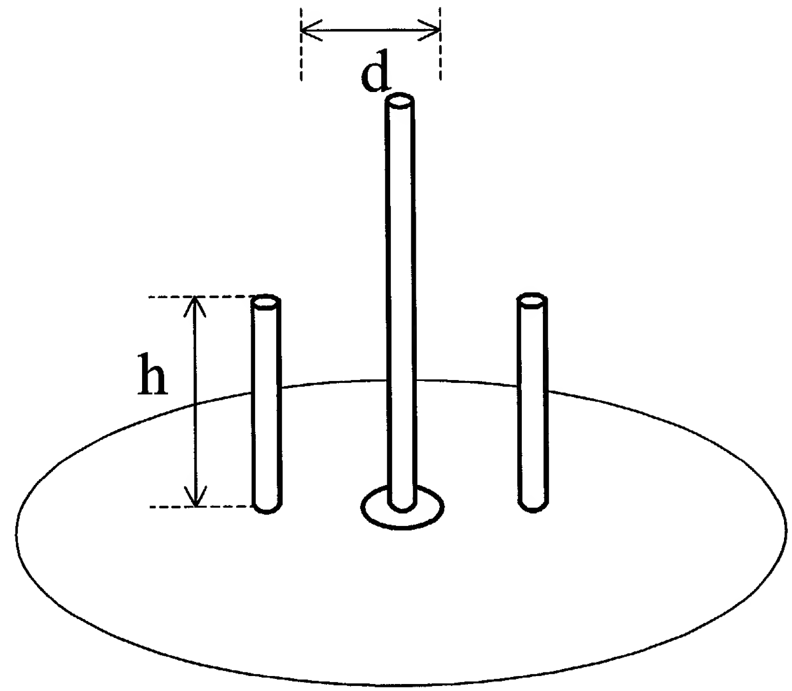
The above genetic algorithm driver was used to search for optimum parameter values for the geometry.

Structures Modeled



Straight Wire with Two Parasites

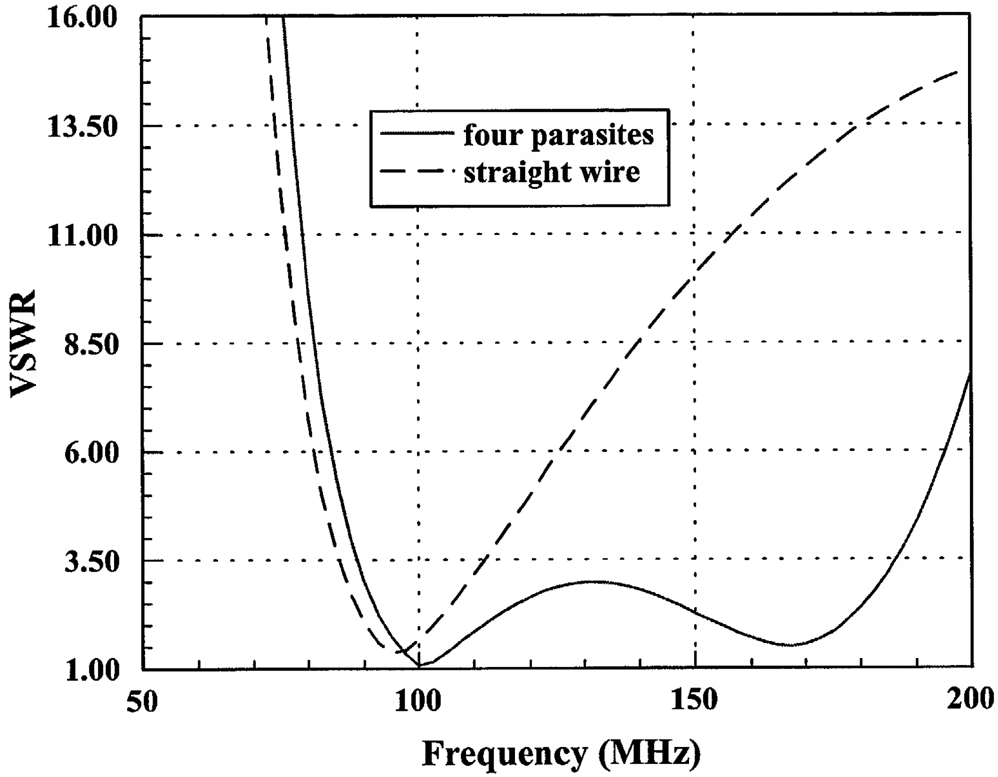


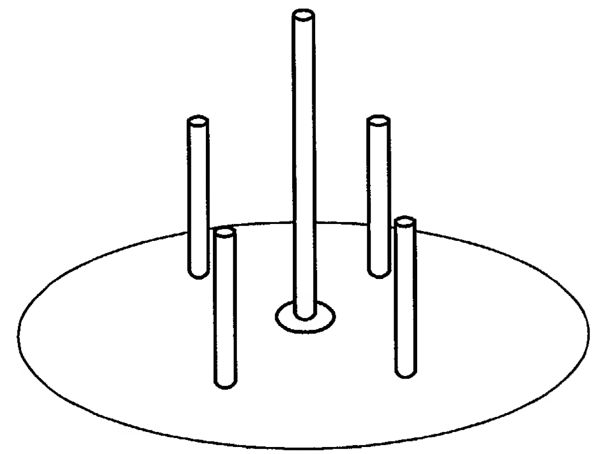


Bandwidth: VSWR < 3.5
f2/f1 : 1

Single Wire	Sleeve Antenna
f1 = 85 MHz	f1 = 90 MHz
f2 = 112 MHz	f2 = 172 MHz
1.32: 1	1.9:1

Straight Wire with Four Parasites



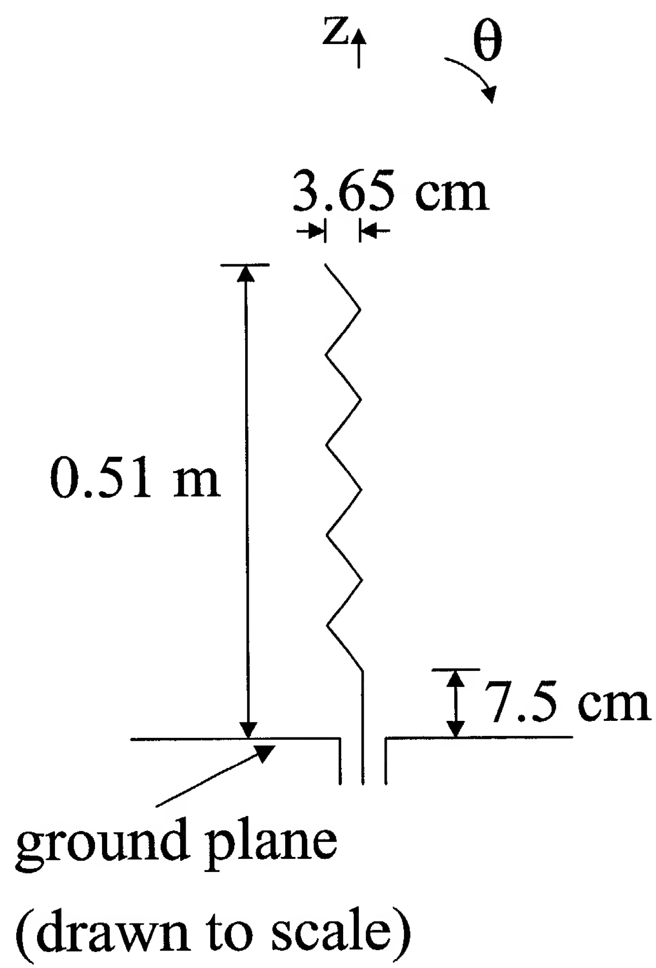


f1 = 90 MHz
f2 = 185 MHz
2.05:1

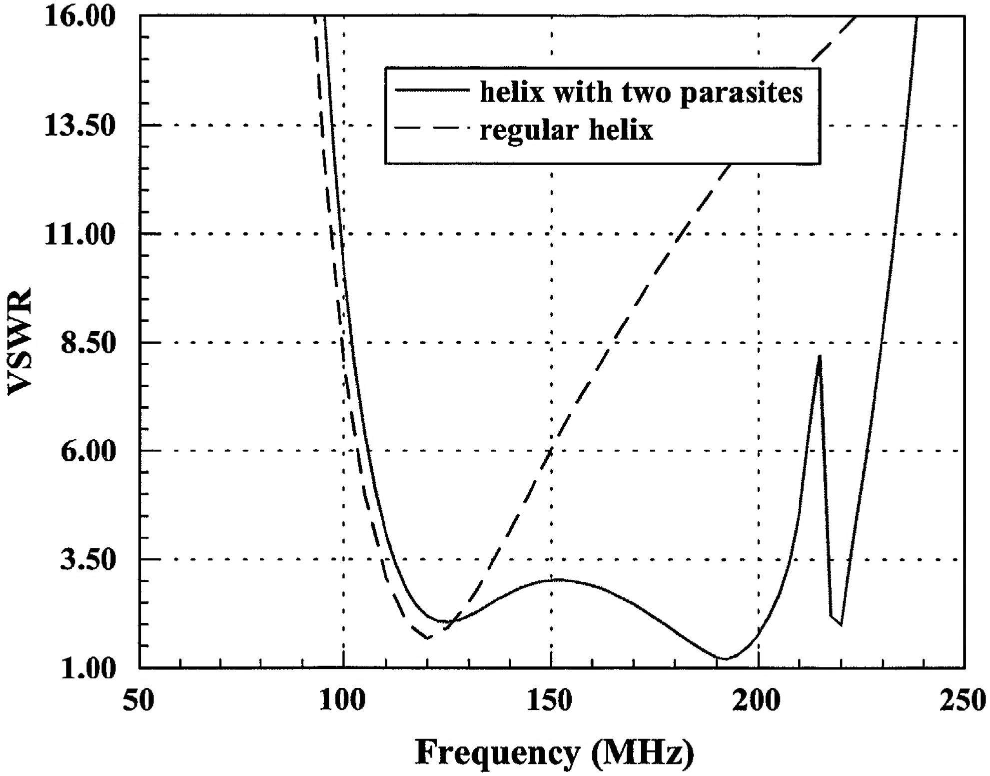
Basic Geometry of Helical Antenna

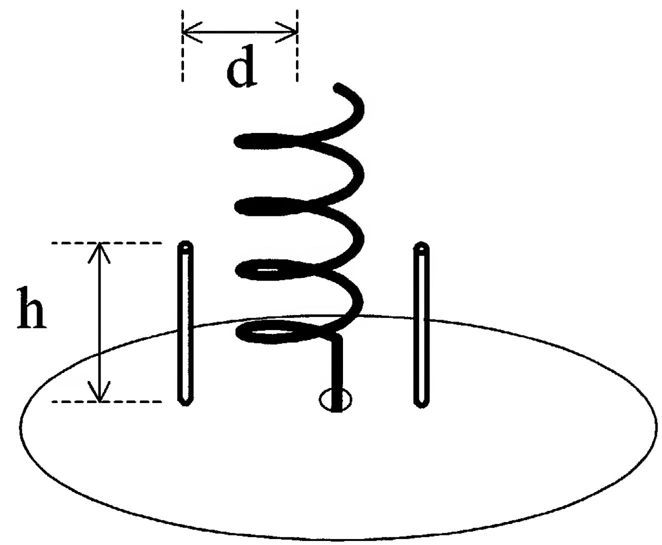
Helix Geometry

Height of straight wire	7.5 cm
Circumference of one turn	15 cm
Total wire length	75 cm
Number of turns	4.5
Pitch angle	40 degrees
Wire diameter	1 cm



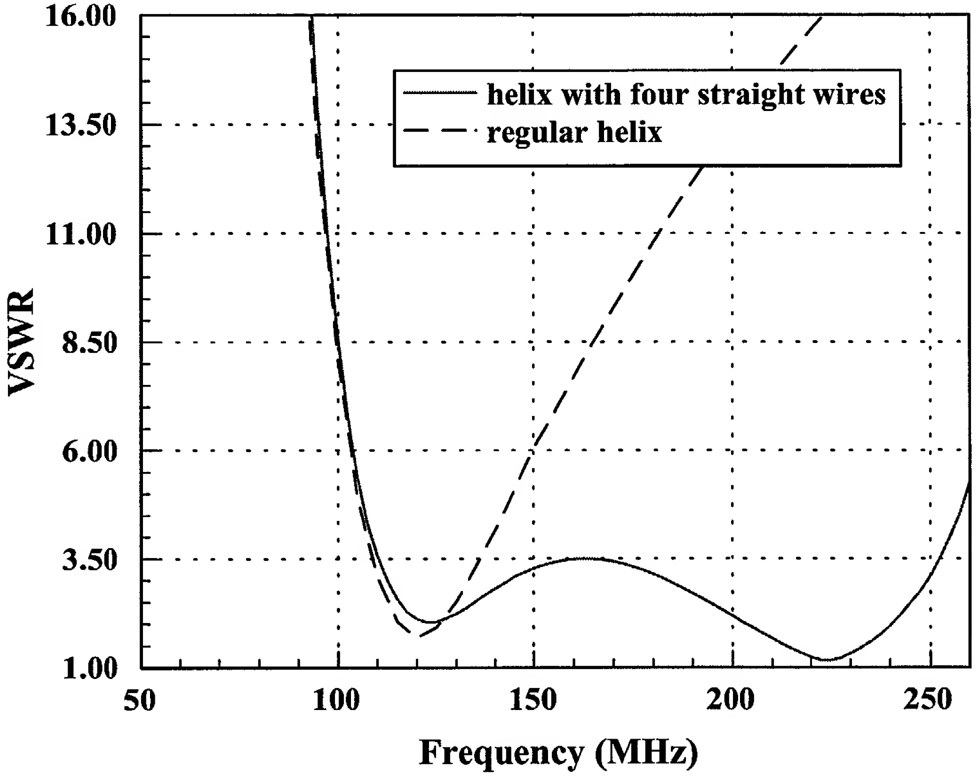
Helix with Two Straight Wire Parasites

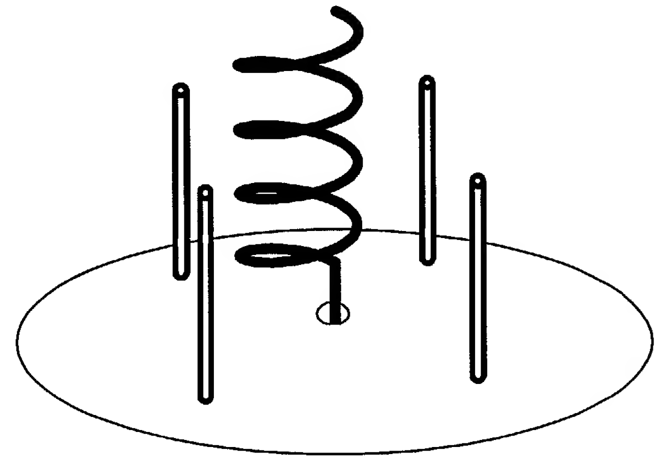




f1 = 112 MHz
f2 = 208 MHz
1.86:1

Helix with Four Straight Wire Parasites

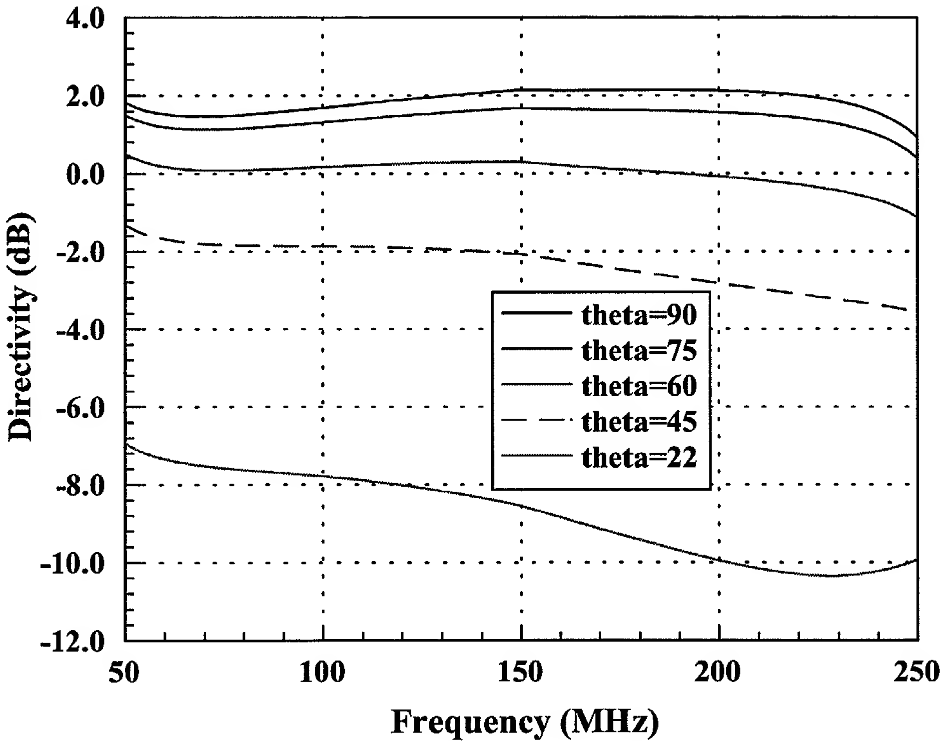




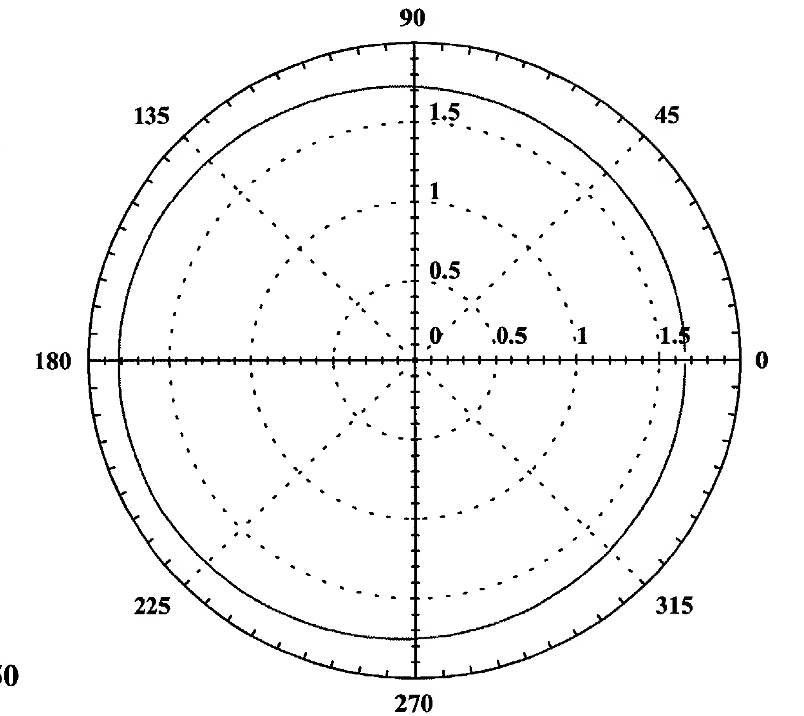
f1 = 112 MHz
f2 = 250 MHz
2.23:1

Helix with Four Straight Wire Parasites

Directivity vs θ for ϕ = 0

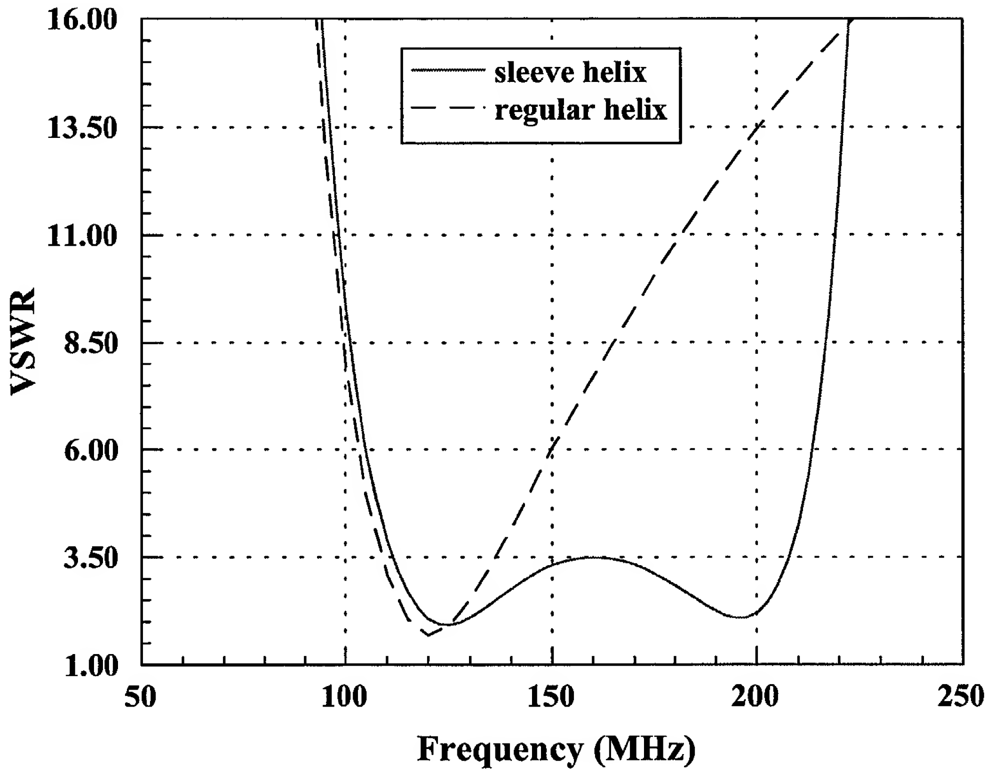


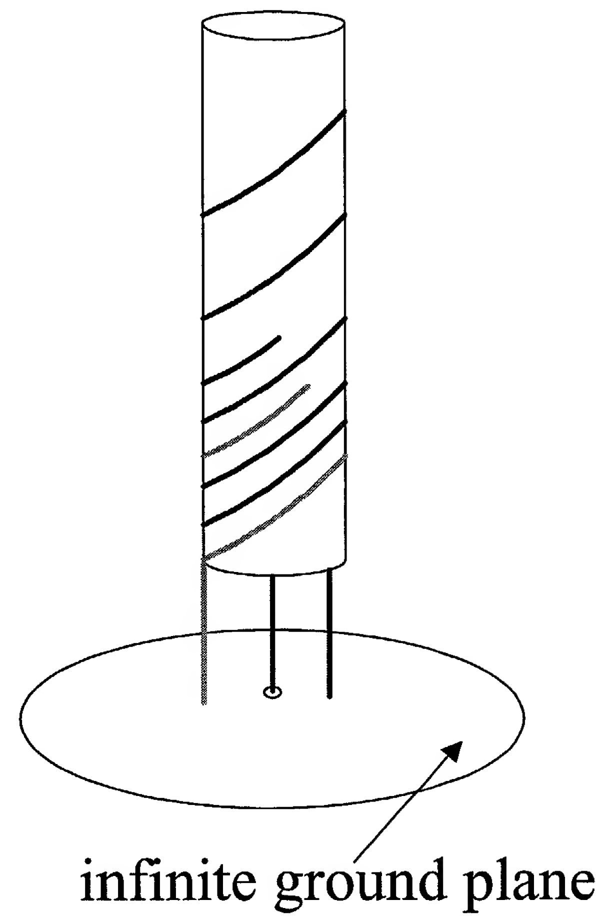
Directivity in H-Plane vs ϕ



f= 190 MHz
(θ = 90)

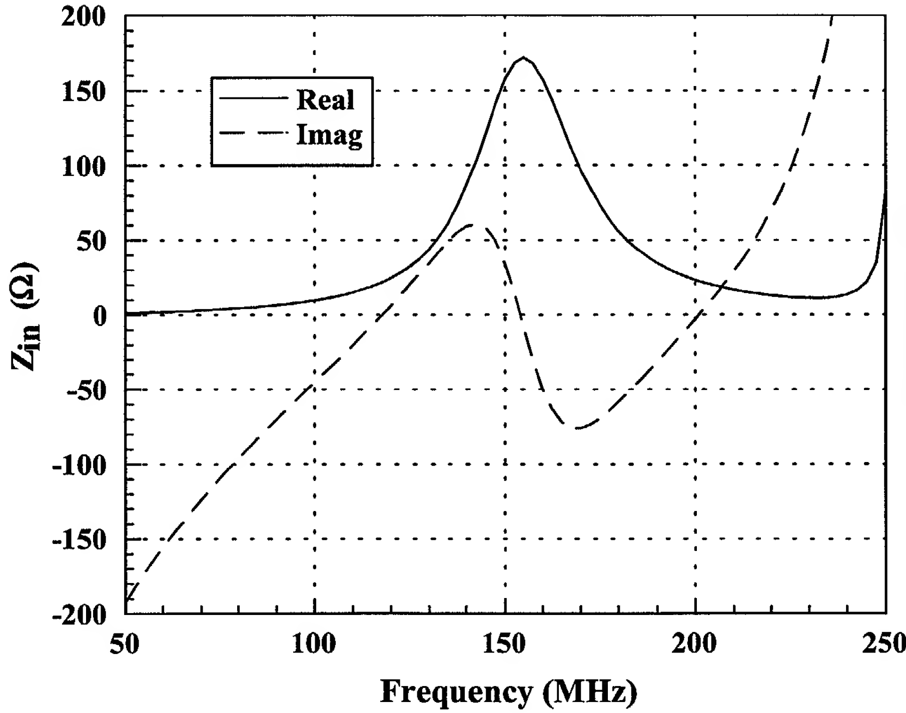
Helix with Two Helical Parasites

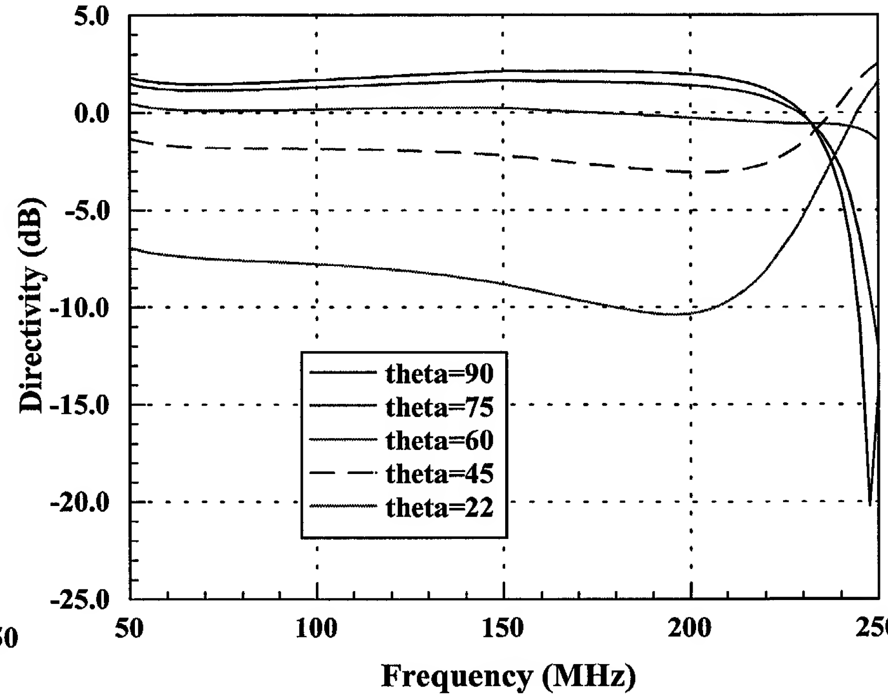




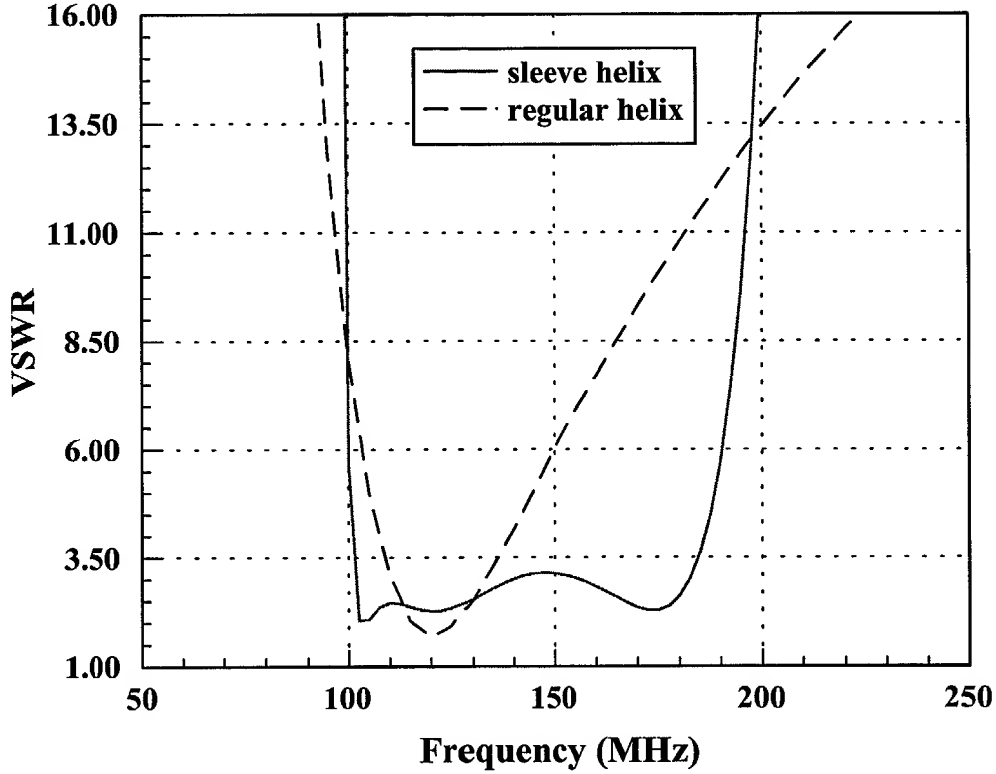
f1 = 112 MHz
f2 = 208 MHz
1.86:1

Helix with Two Helical Parasites

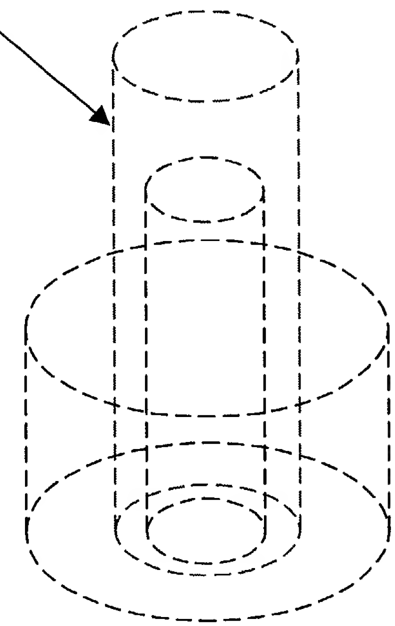




Helix with Inner and Outer Helical Parasites

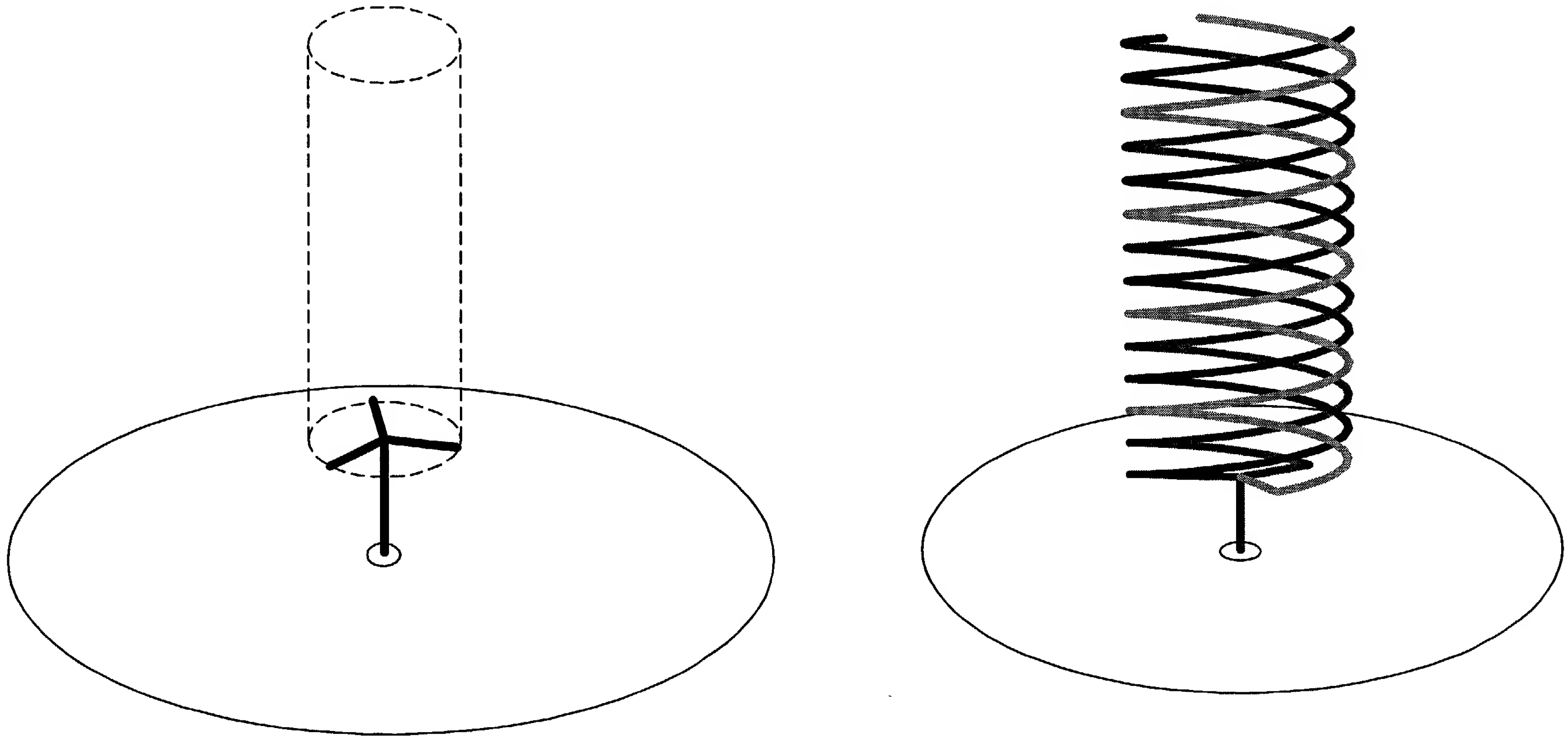


Driven helix wrapped
on middle cylinder

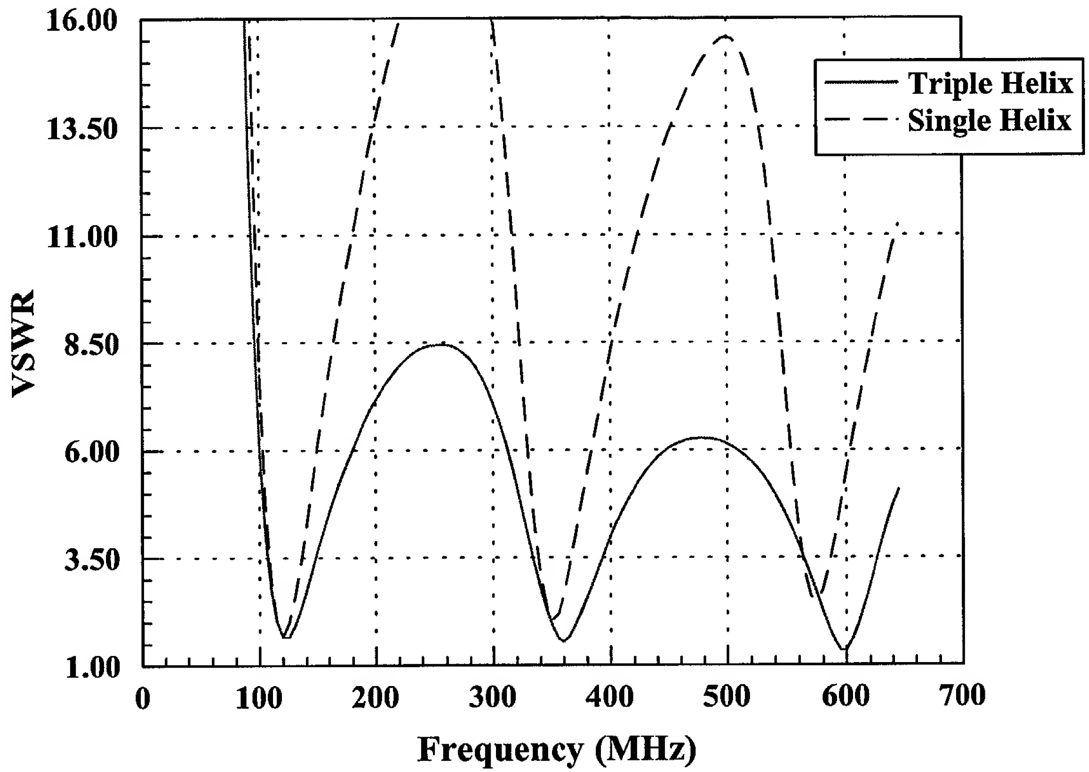


f1 = 101 MHz
f2 = 182.5 MHz
1.81:1

Triple Helix



VSWR of Triple Helix

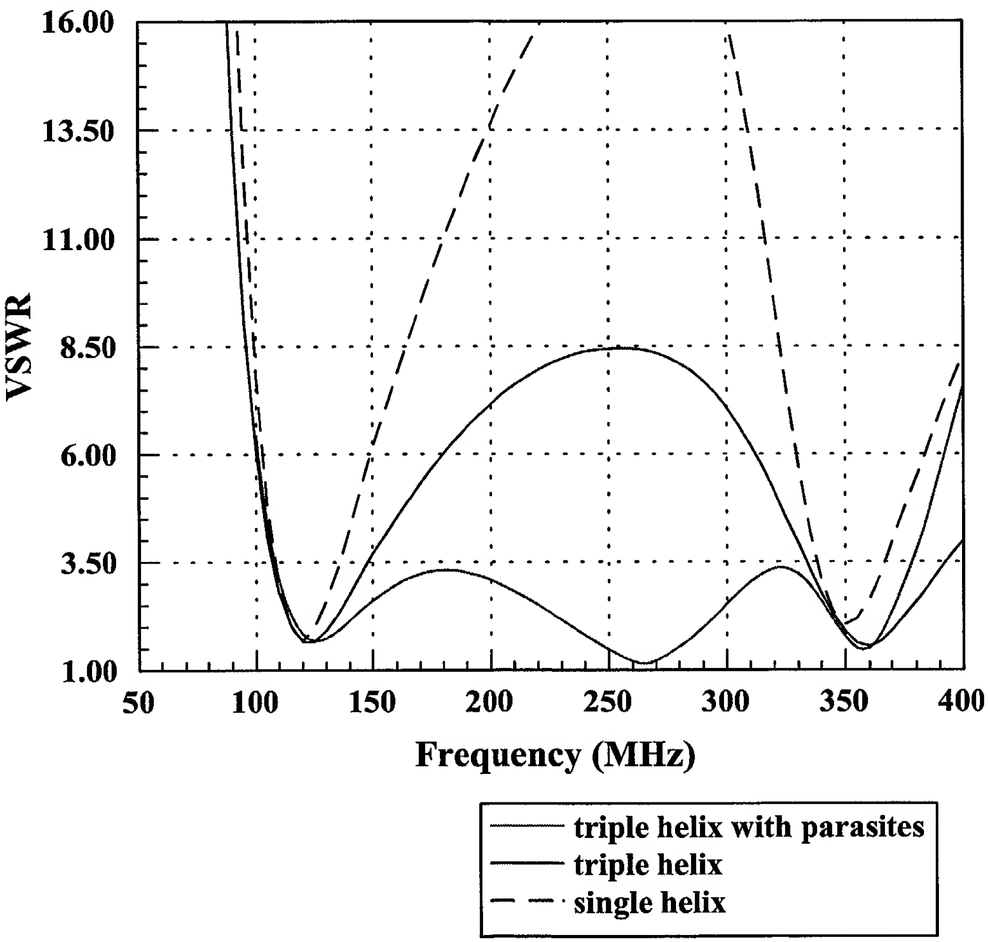


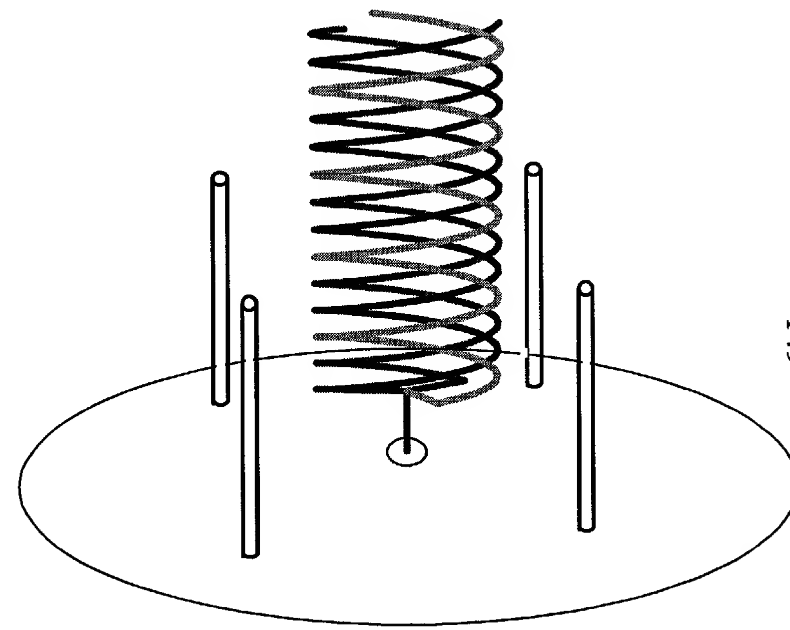
Triple helix has 13.5 turns
(4.5 for each helix)

To represent geometry:
675 Unknowns
(25 points/turn)

To represent current:
150 Unknowns
(25 unknowns/λ
at 400 MHz)

Triple Helix with Four Straight Wire Parasites

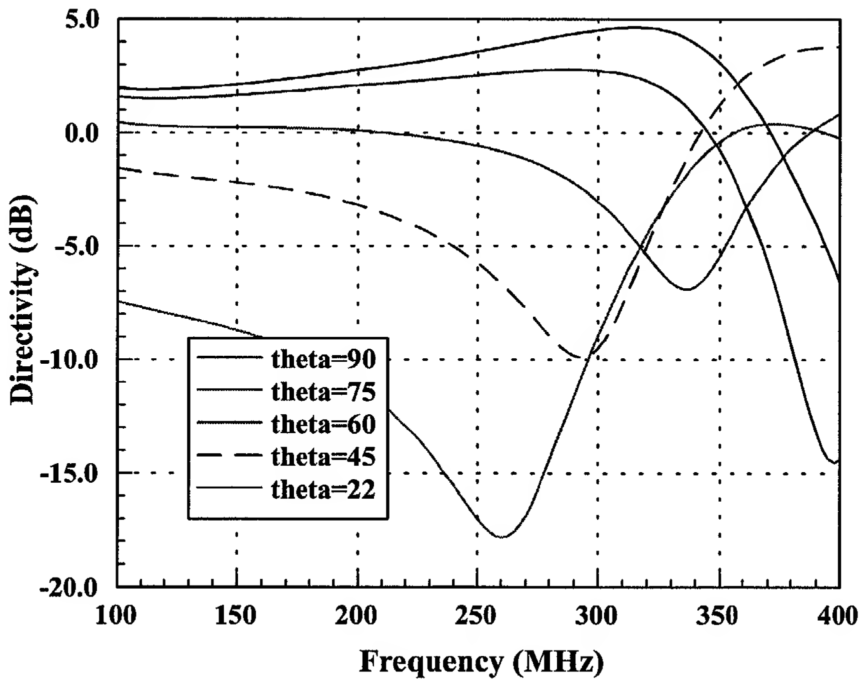




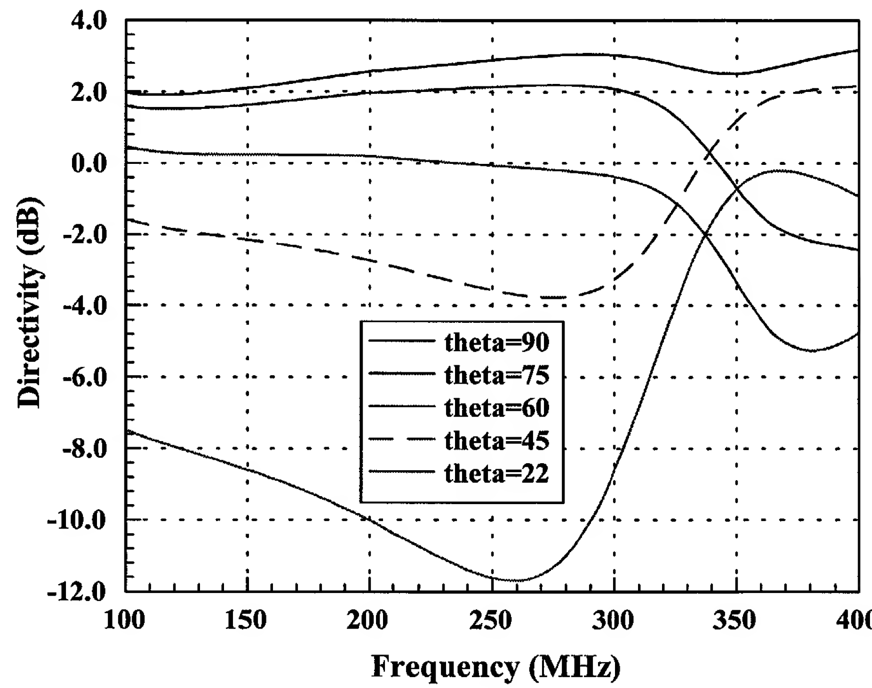
f1 = 110 MHz
f2 = 380 MHz
3.45:1

Triple Helix with Four Straight Wire Parasites

Triple Helix



Triple Helix with Parasites



Summary of Results

Driven Element	Number of Parasites	Type of Parasites	Bandwidth Ratio
Straight wire	2	Straight wire	1.90:1
Straight wire	4	Straight wire	2.05:1
Helix	2	Straight wire	1.86:1
Helix	4	Straight wire	2.23:1
Helix	2	Helix (same cylinder)	1.86:1
Helix	2	Helix (different cylinders)	1.81:1
Triple helix	4	Straight wire	3.45:1

Conclusions

Parasitic straight wires and helices are useful for improving the bandwidth of helical antennas.

The triple helix has a reduced VSWR over the frequency band which makes the structure more amenable to improvement by parasites.

The following further portion of this disclosure was also created in *Powerpoint* for purposes of further describing the present invention. It particularly concerns the sleeve-cage monopole and sleeve helix for wide band operation. It sets forth the objectives, considerations, and questions addressed in the development of the present invention, and presents relevant background information and illustrations of the antennas discussed within.

The VSWR, input impedance, and directivity are given for each antenna with and without the addition of the parasitic elements. Illustrations and graphical data for the cage monopole, the sleeve-cage monopole, the quadrifilar helix, and the sleeve-helix are presented. The relevant data for each is then shown in a table so that a side-by-side comparison can be made to clearly illustrate the improvements made in the antenna characteristics by the optimal placement of parasitic elements.

The physical measurements and characterization values of various antennas optimized for various VSWR values are presented. This data is then presented in a comparison to several background antennas to illustrate the improvements in antenna performance made by the present invention.

Overview

- Introduction
- Literature background
- Procedure for optimization of antennas with parasitic elements
- Description of measurements
- Results (measured and computed)
- Conclusions

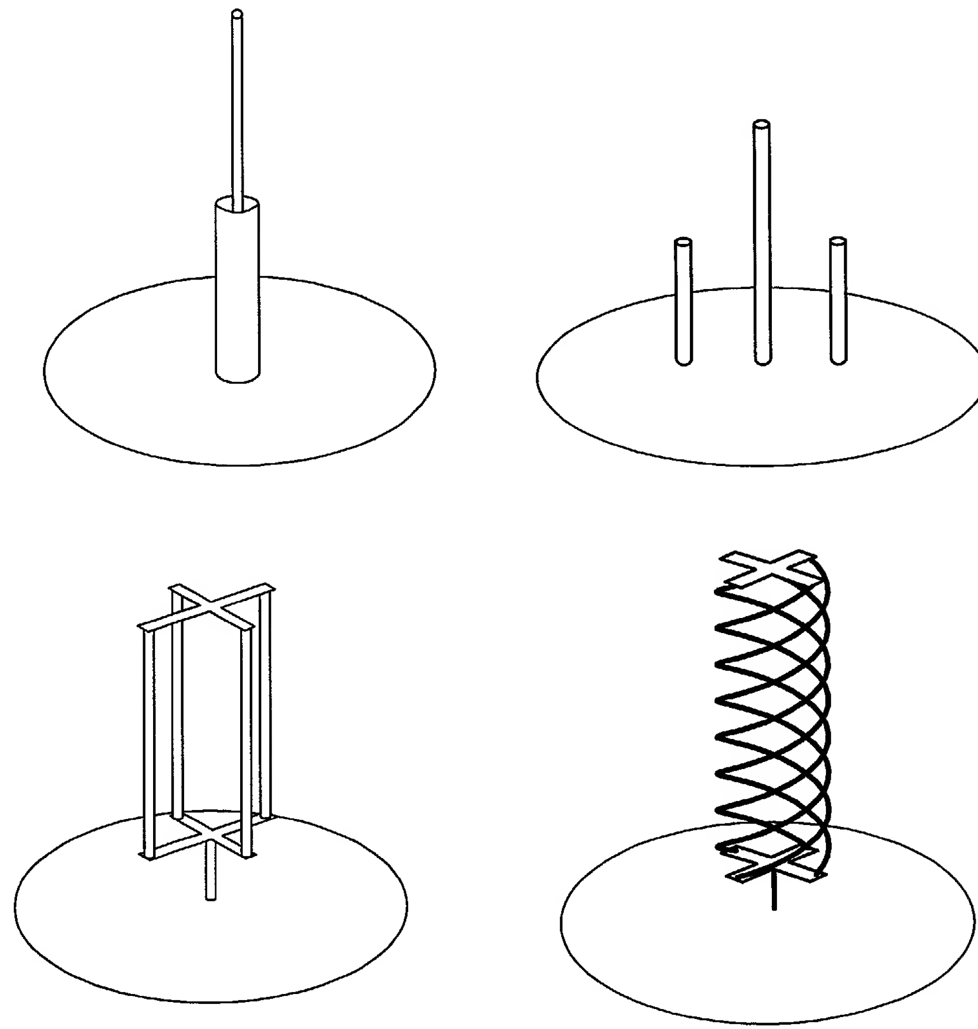
Introduction

Objectives for Antenna
- Low-profile
- Omnidirectional
- Broadband

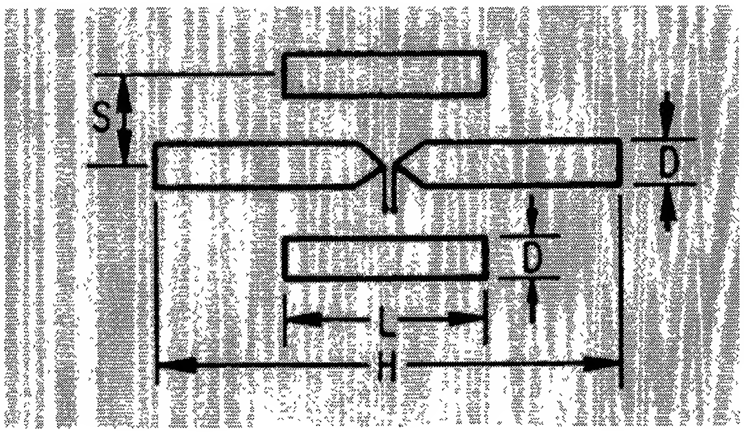
Approach
- Sleeve antennas
- Cage antennas
- Helix for reduced height

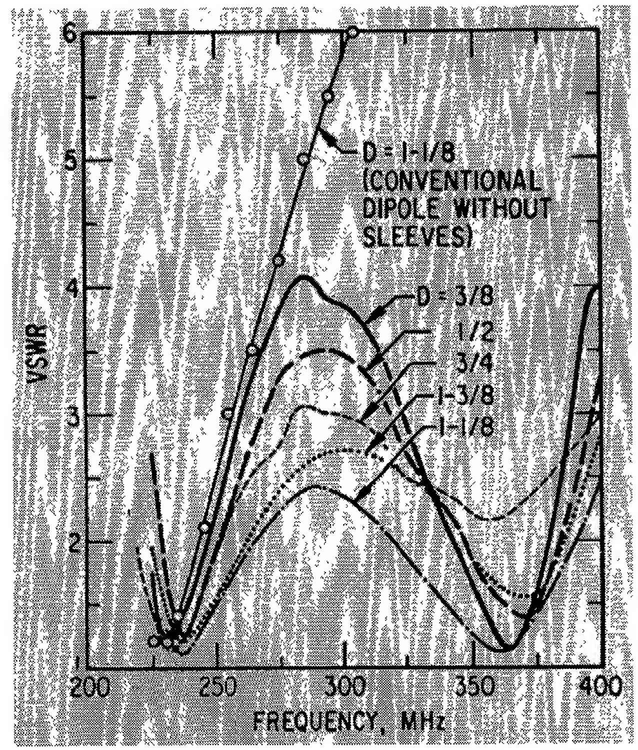
Bandwidth

Ratio	**Percent**
$\dfrac{f_1}{f_2}$	$100\dfrac{(f_1 - f_2)}{\sqrt{f_1 f_2}}$



Literature: Open Sleeve Dipole



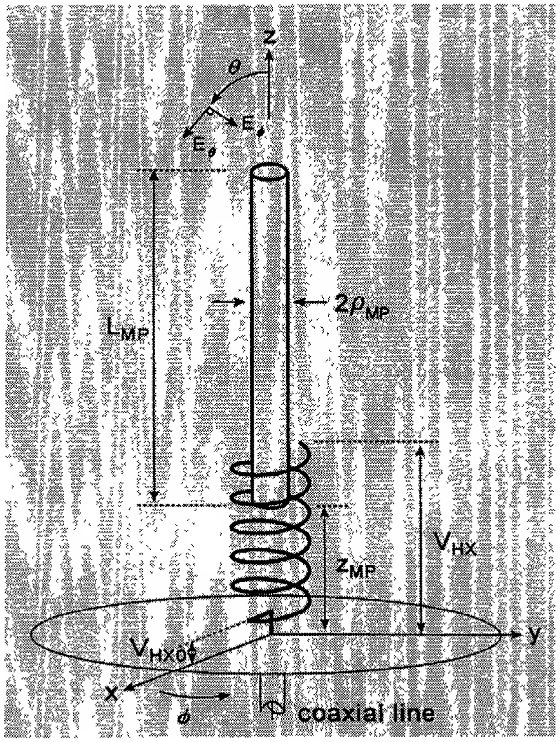


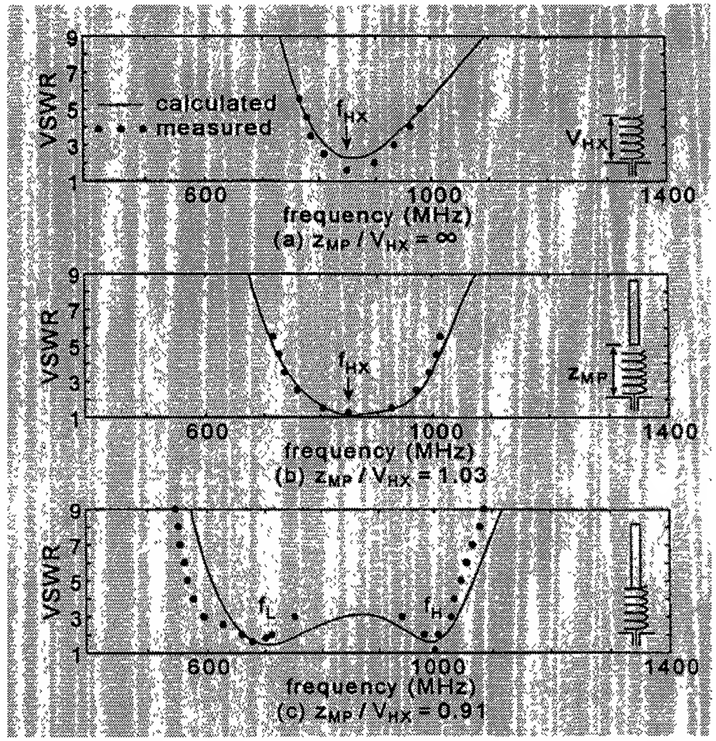
VSWR < 2.5   225 - 400 MHz
BW 1.77,   58.3%

D = 1.125”   H= 20.2”   L = 11.38”   S = 2”

H.E. King and J.L. Wong, “An Experimental Study of a Balun-Fed Open-Sleeve Dipole in Front of a Metallic Reflector,” *IEEE Trans. Antennas Propagat.* (Commun.), vol. AP-20, pp. 201-204, March 1972.

Literature: Helix with Parasitic Monopole





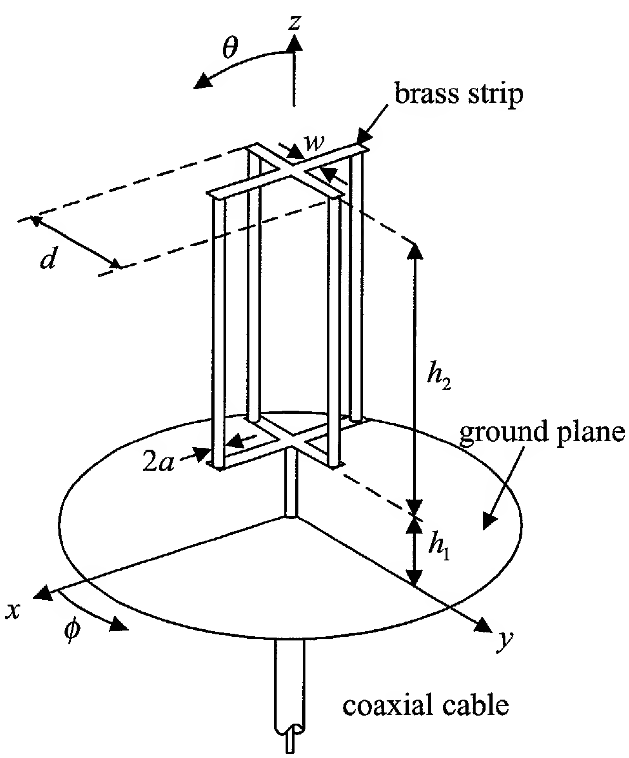
VSWR < 2, 785 - 961 MHz
BW 1.22, 20%

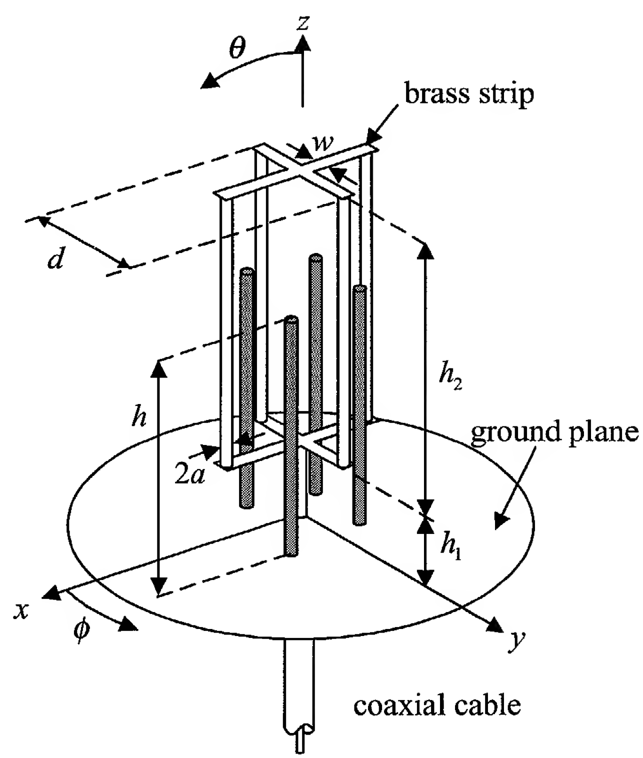
VSWR < 2, 662 - 757 MHz
BW 1.14, 13.4%

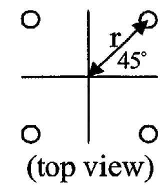
VSWR < 2, 957 - 1014 MHz
BW 1.05, 5.78%

H. Nakano, et.al., "Realization of Dual-Frequency and Wide-Band VSWR Performances Using Normal-Mode Helical and Inverted-F Antennas," *IEEE Trans. Antennas Propagat.* vol. AP-46, pp. 788-793, June 1998.

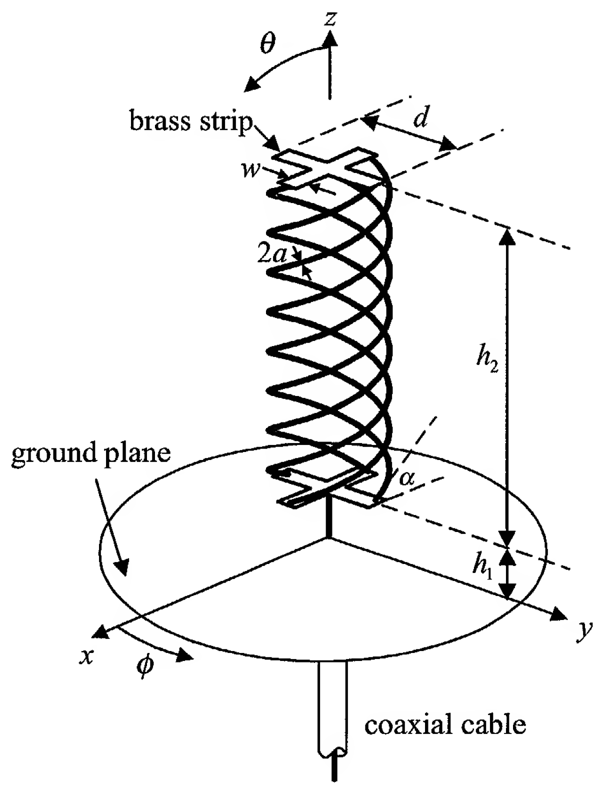
Cage and Sleeve-Cage Monopole

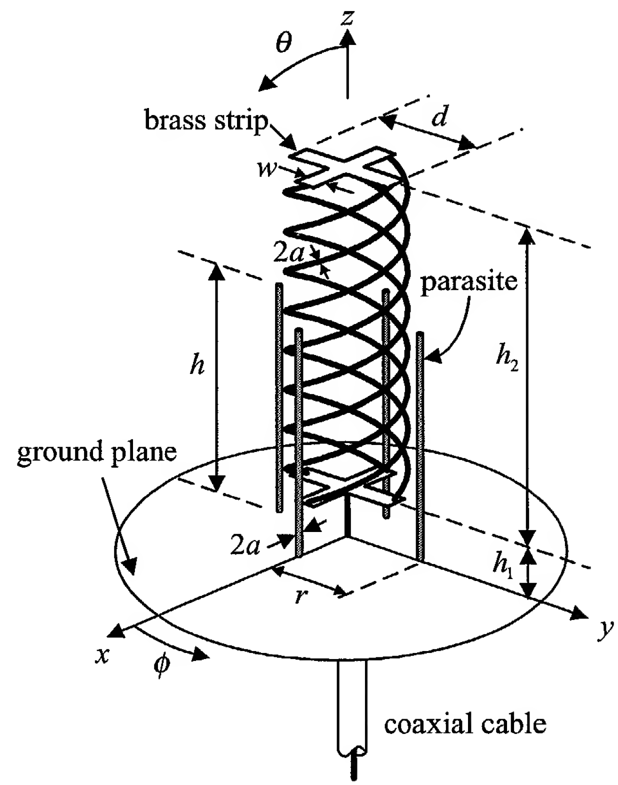




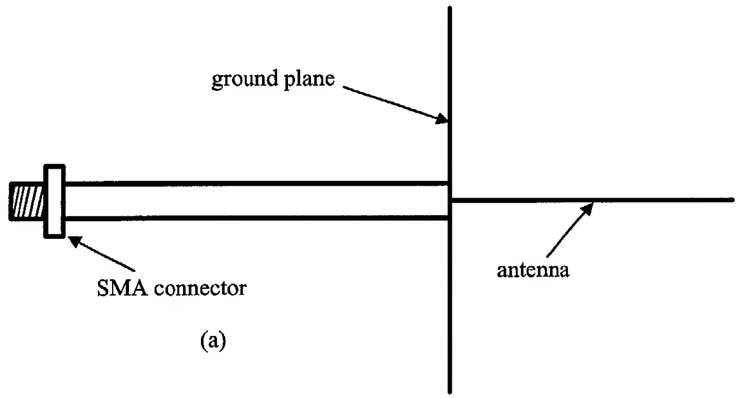


Quadrifilar Helix and Sleeve Helix

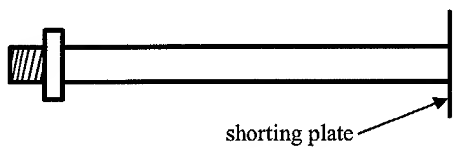




Measuring Input Impedance

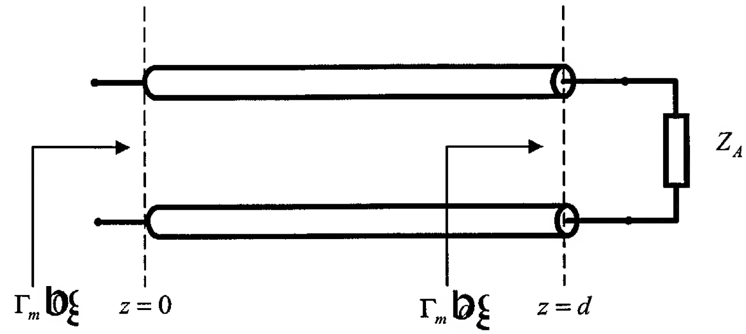


(a)

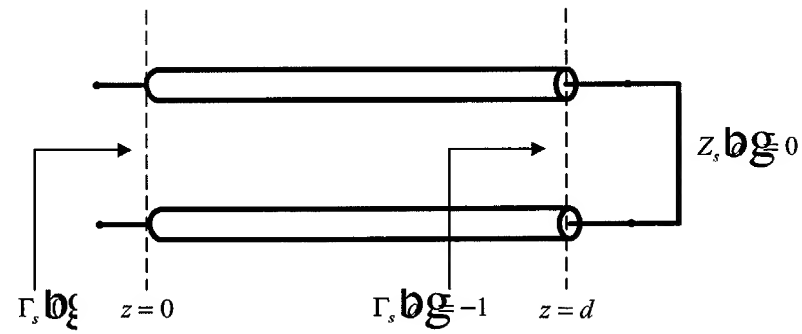


(b)

$$Z_m(d) = Z_0 \frac{\Gamma_s(0) - \Gamma_m(0)}{\Gamma_s(0) + \Gamma_m(0)}$$

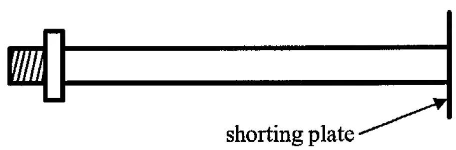


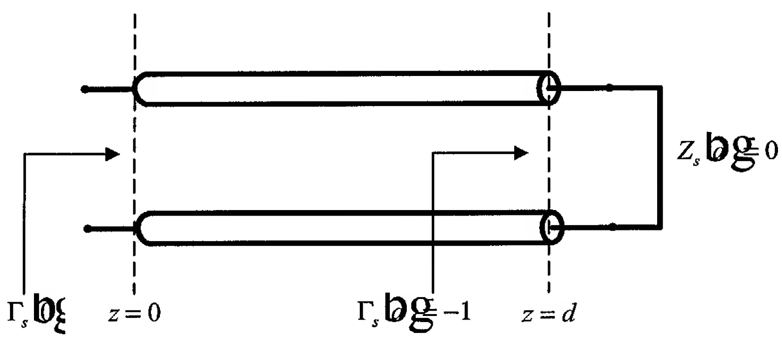
(a)

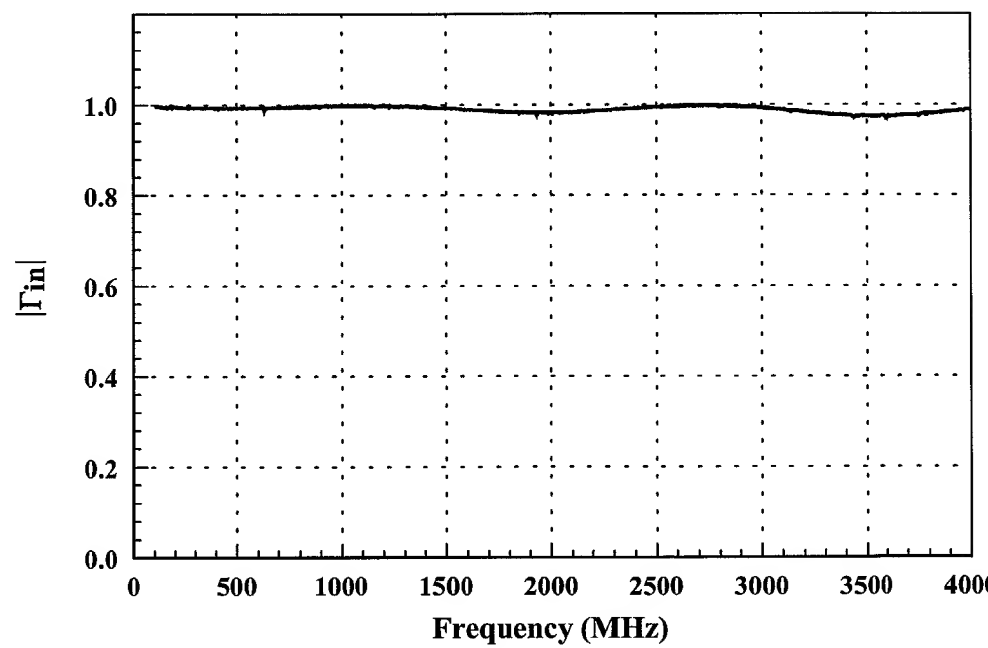


(b)

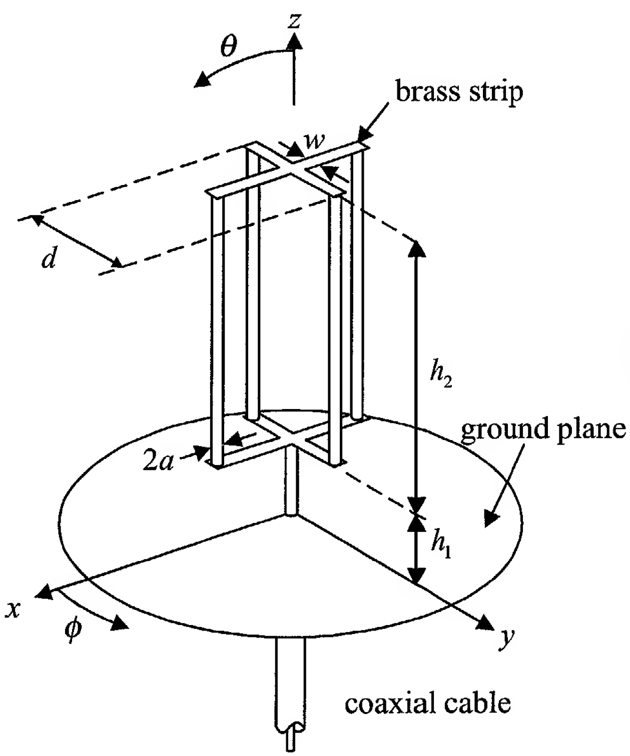
Reflection Coefficient Magnitude for Short

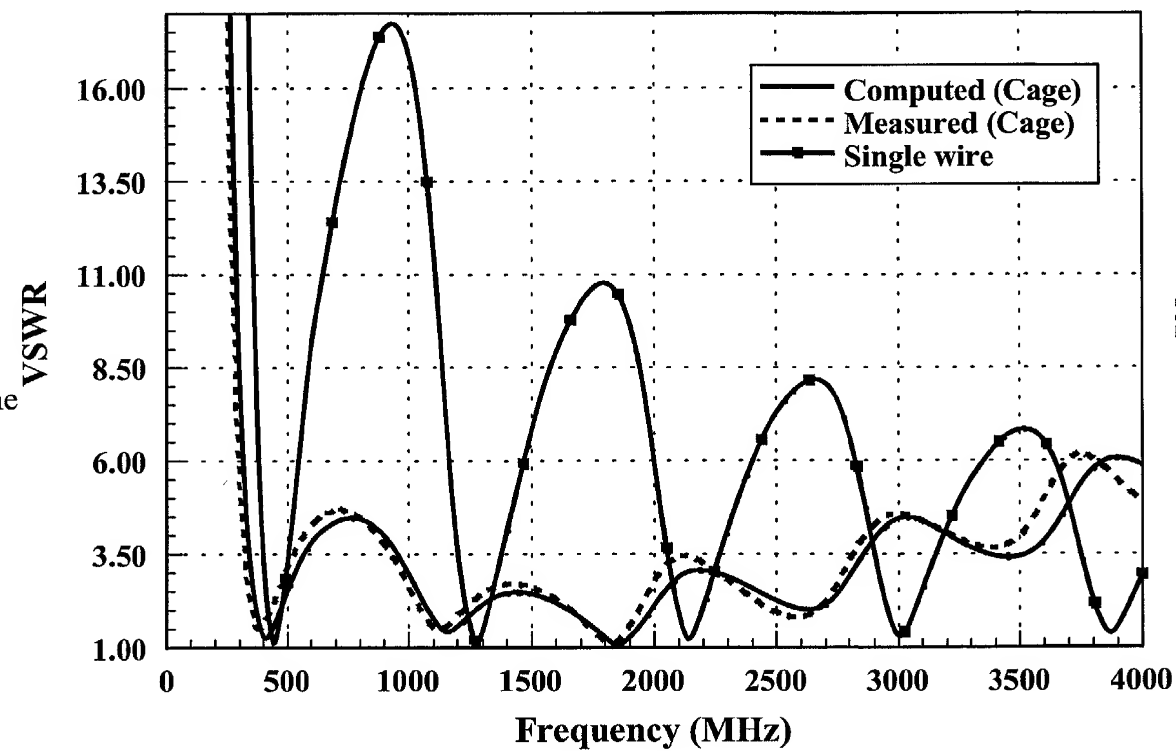






VSWR of Cage Monopole

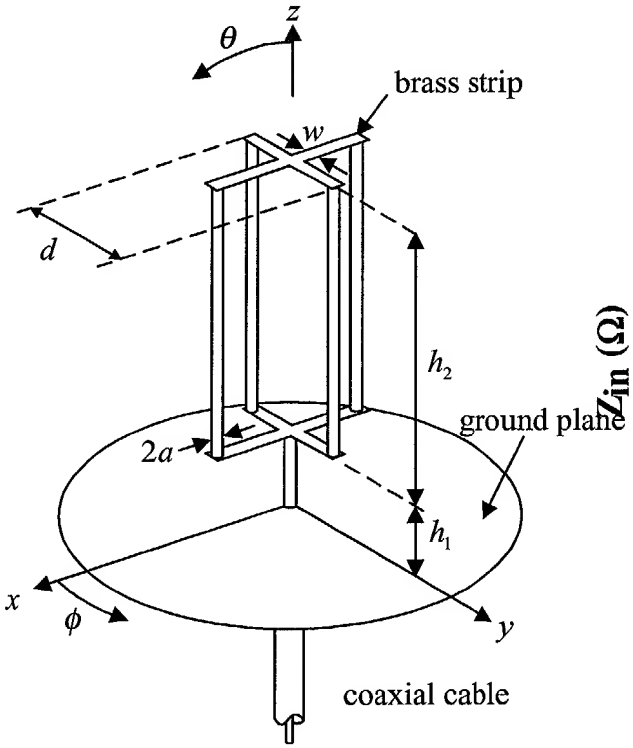


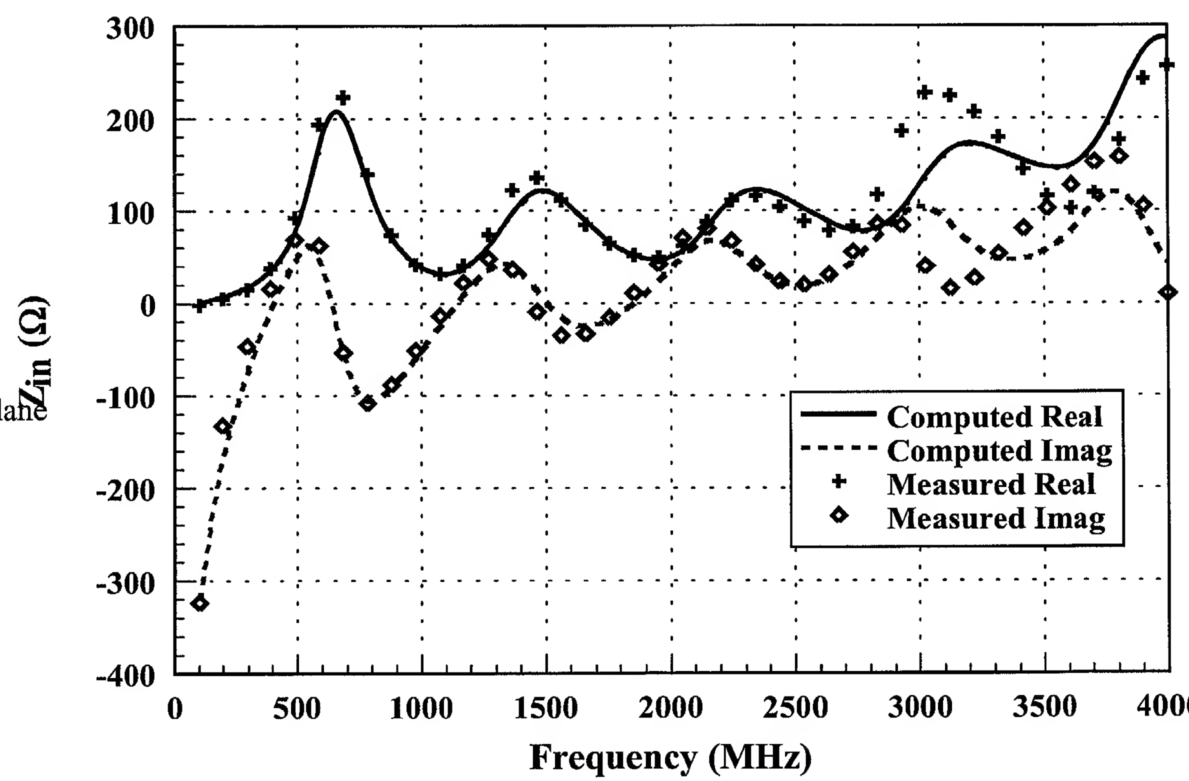


$a = 0.814$ mm, d= 2.2 cm, w= 3.256mm, h_1 =1.2 cm, h_2 =16 cm.

VSWR < 5.0, 300-3700 MHz
BW 12.3:1

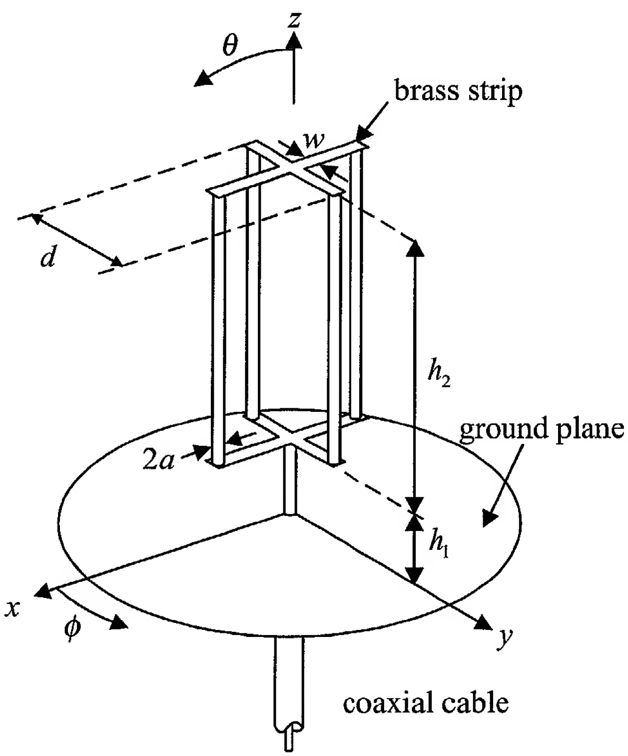
Input Impedance of Cage Monopole

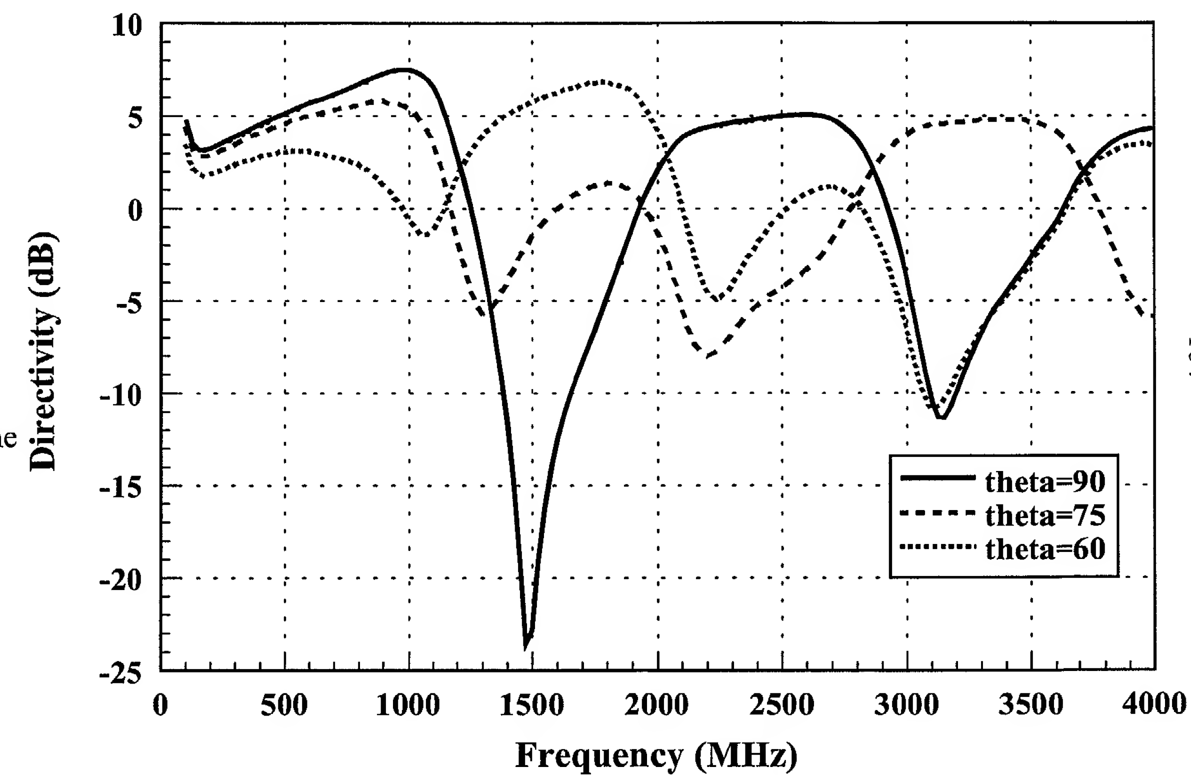




$a = 0.814$ mm, d= 2.2 cm, w= 3.256mm, h_1 =1.2 cm, h_2 =16 cm.

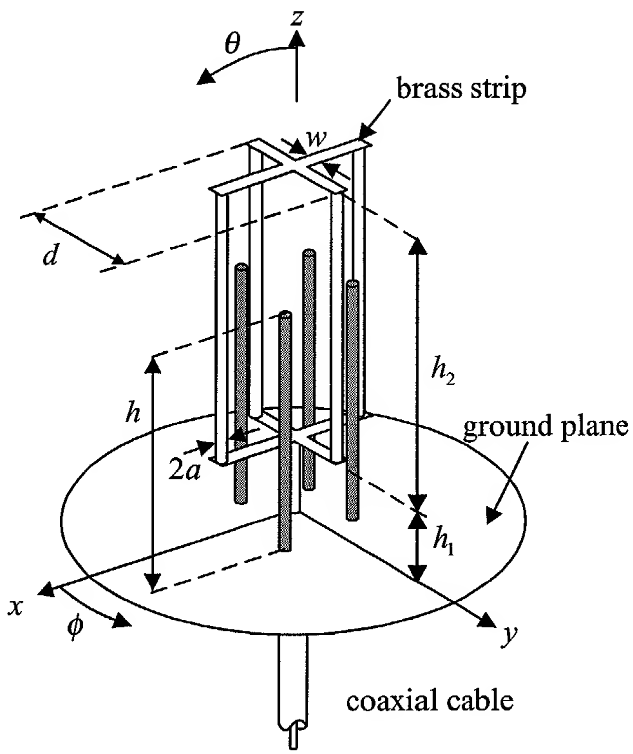
Directivity of Cage Monopole

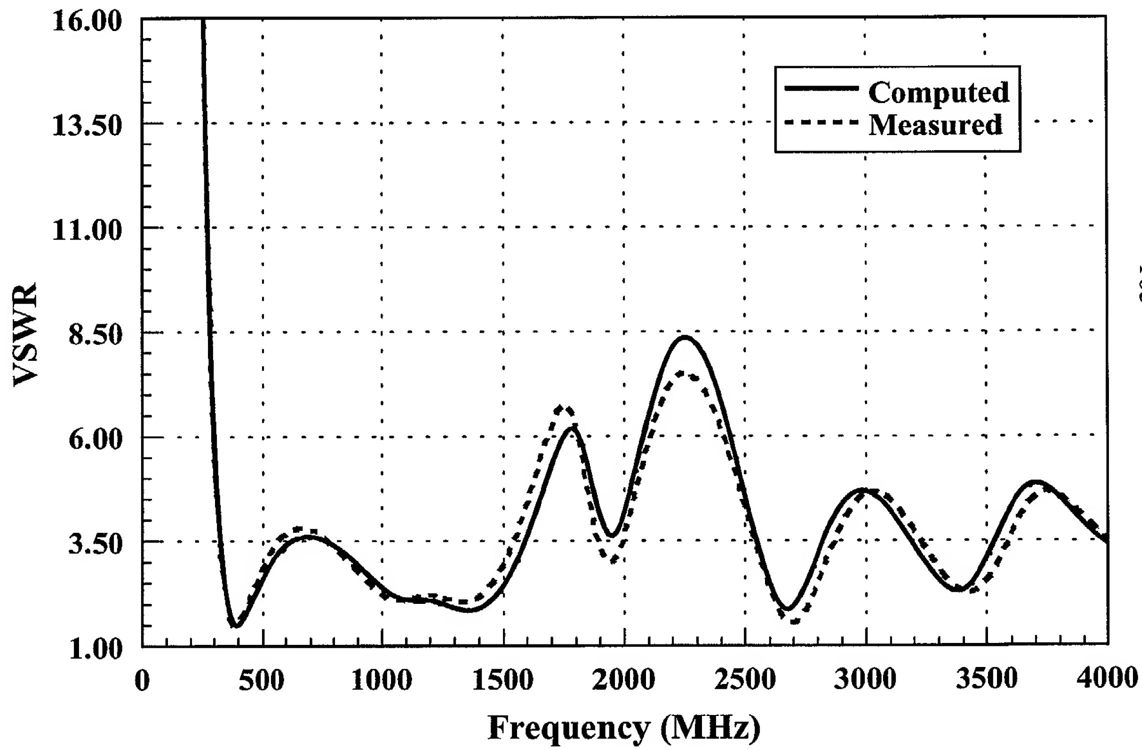




a = 0.814 mm, d= 2.2 cm, w= 3.256mm, h_1 =1.2 cm, h_2 =16 cm.

VSWR of Sleeve-Cage Monopole

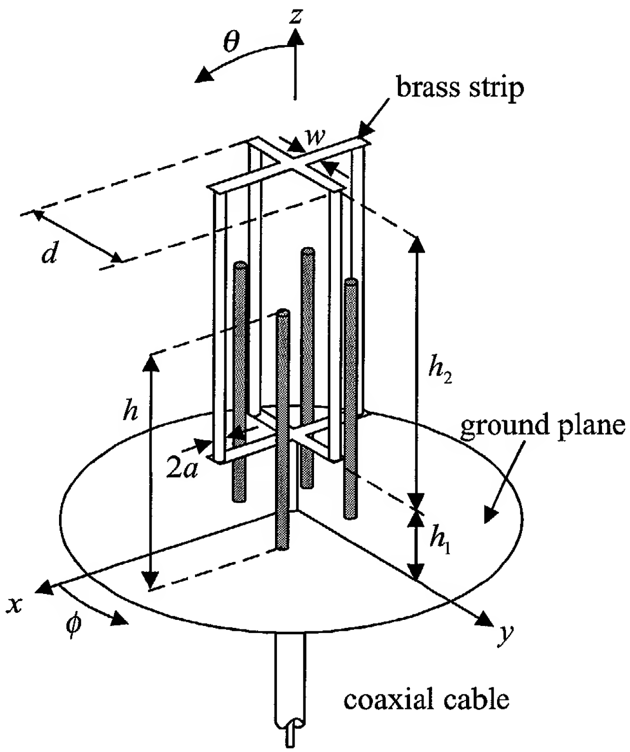


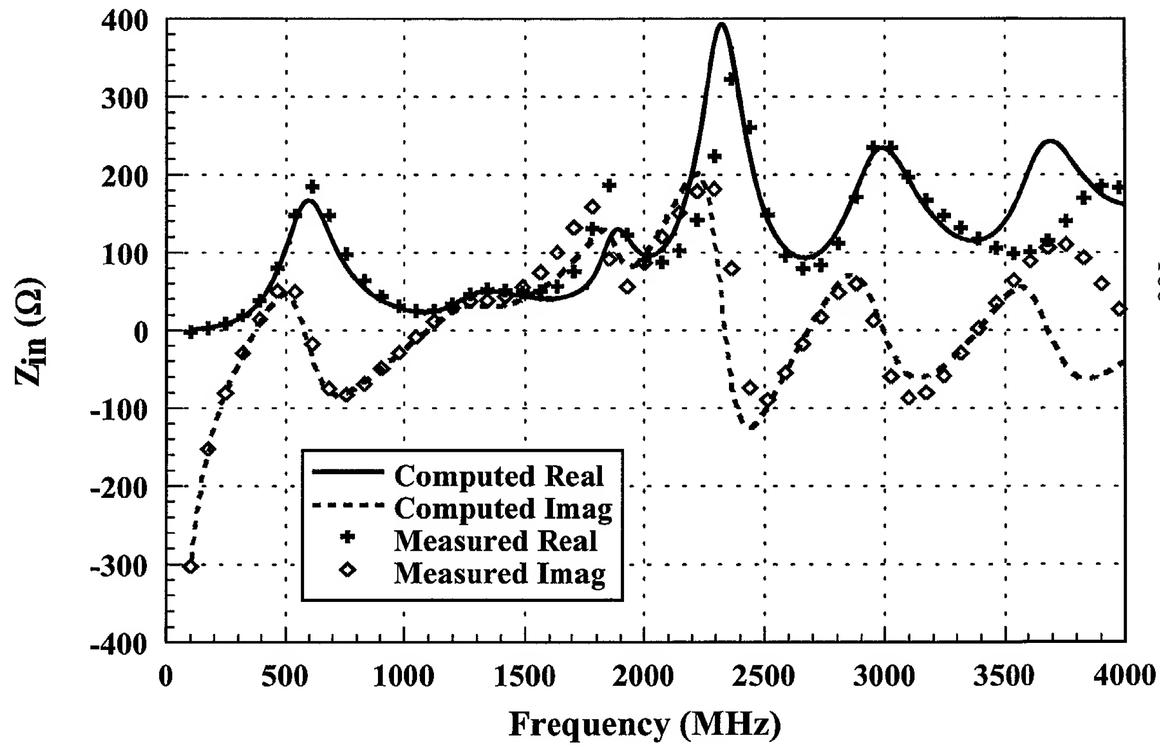


$a = 0.814$ mm, $d = 2.2$ cm, $w = 3.256$mm, $h_1 = 1.2$ cm, $h_2 = 16$ cm, $r = 2.5$ cm, $h = 4$ cm.

VSWR < 3.5, 350-1550 MHz
BW 4.4:1

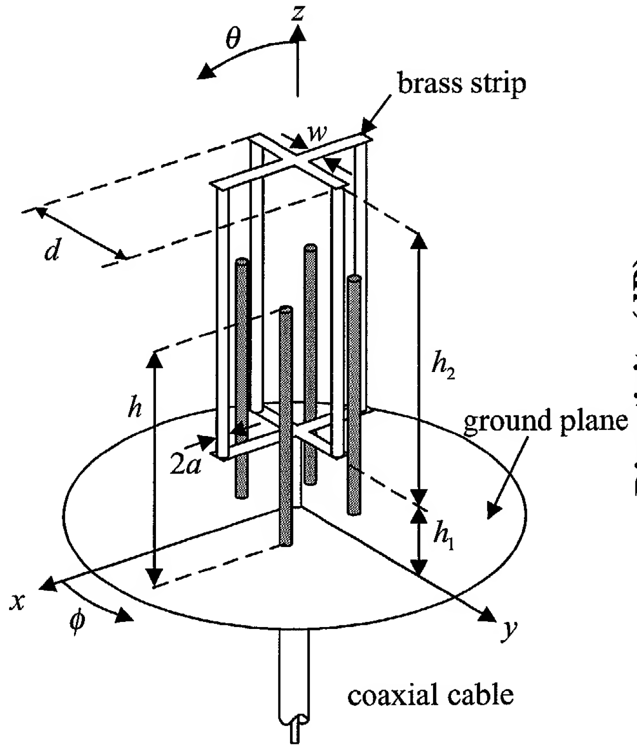
Input Impedance of Sleeve-Cage Monopole

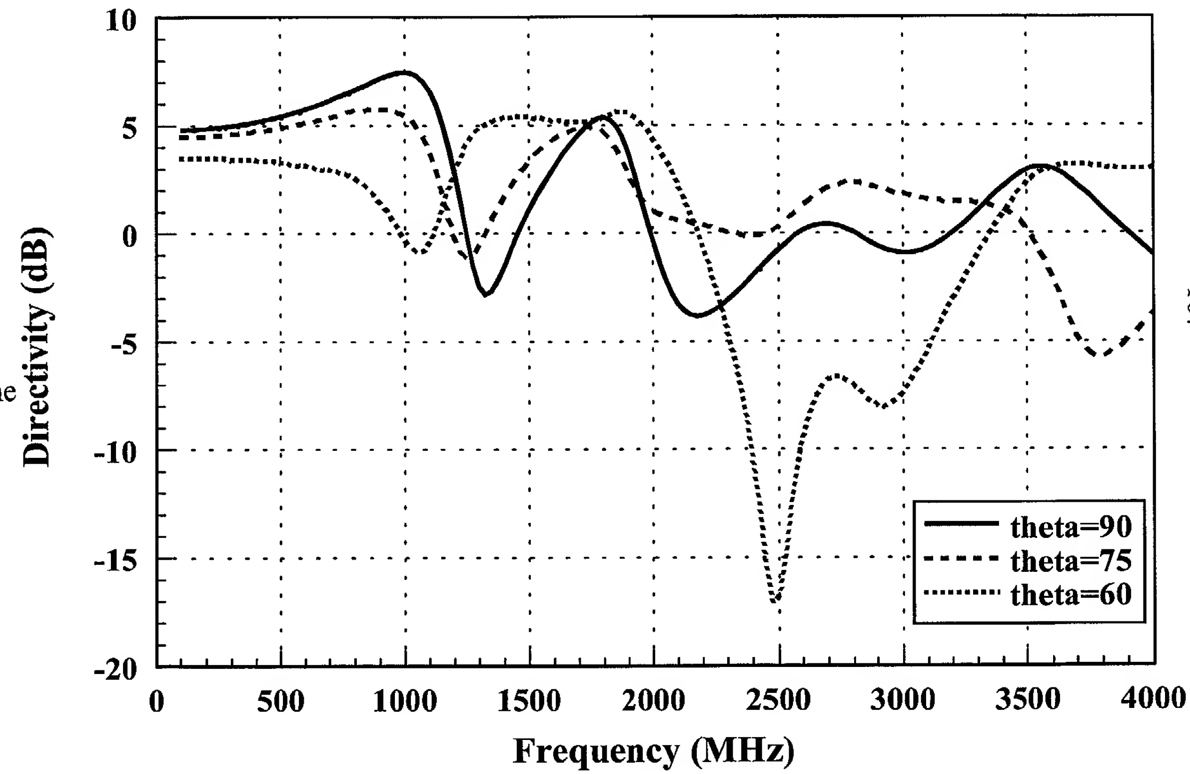




$a = 0.814$ mm, $d = 2.2$ cm, $w = 3.256$mm, $h_1 = 1.2$ cm, $h_2 = 16$ cm, $r = 2.5$ cm, $h = 4$ cm.

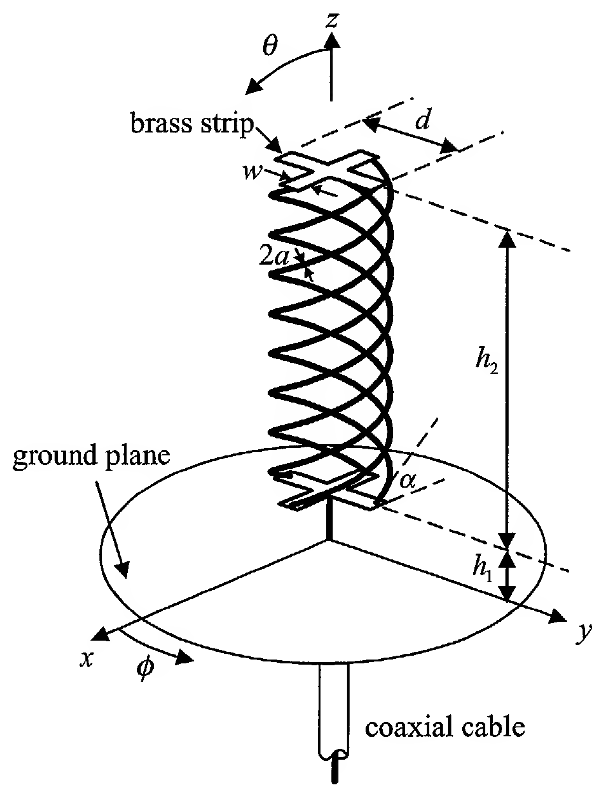
Directivity of Sleeve-cage Monopole

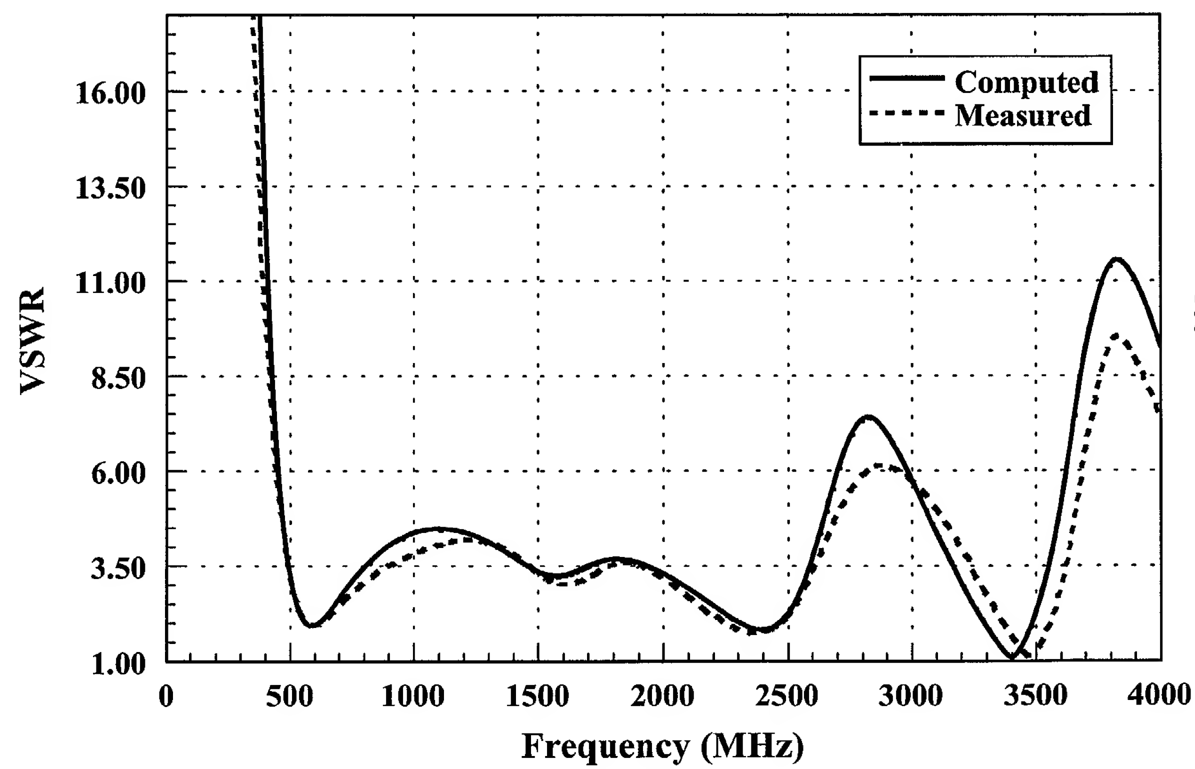




$a = 0.814$ mm, $d = 2.2$ cm, $w = 3.256$mm, $h_1 = 1.2$ cm, $h_2 = 16$ cm, $r = 2.5$ cm, $h = 4$ cm.

VSWR of Quadrifilar Helix

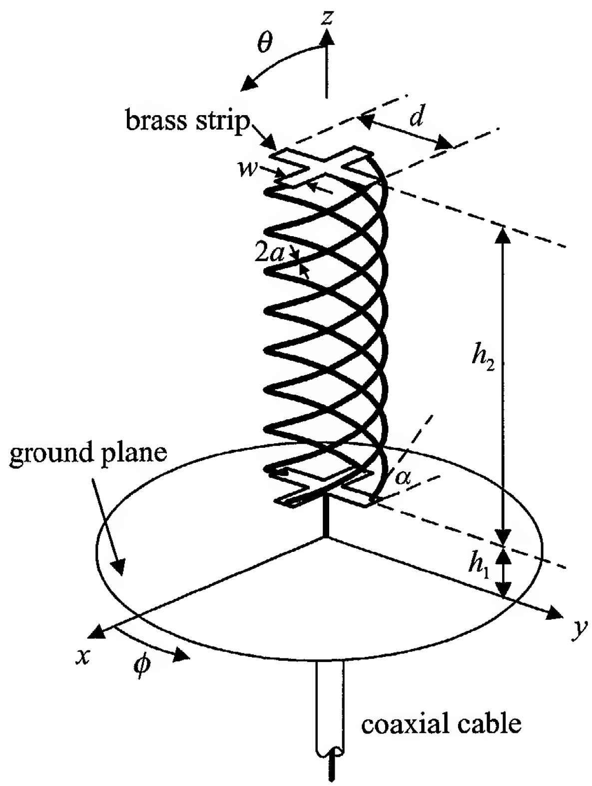


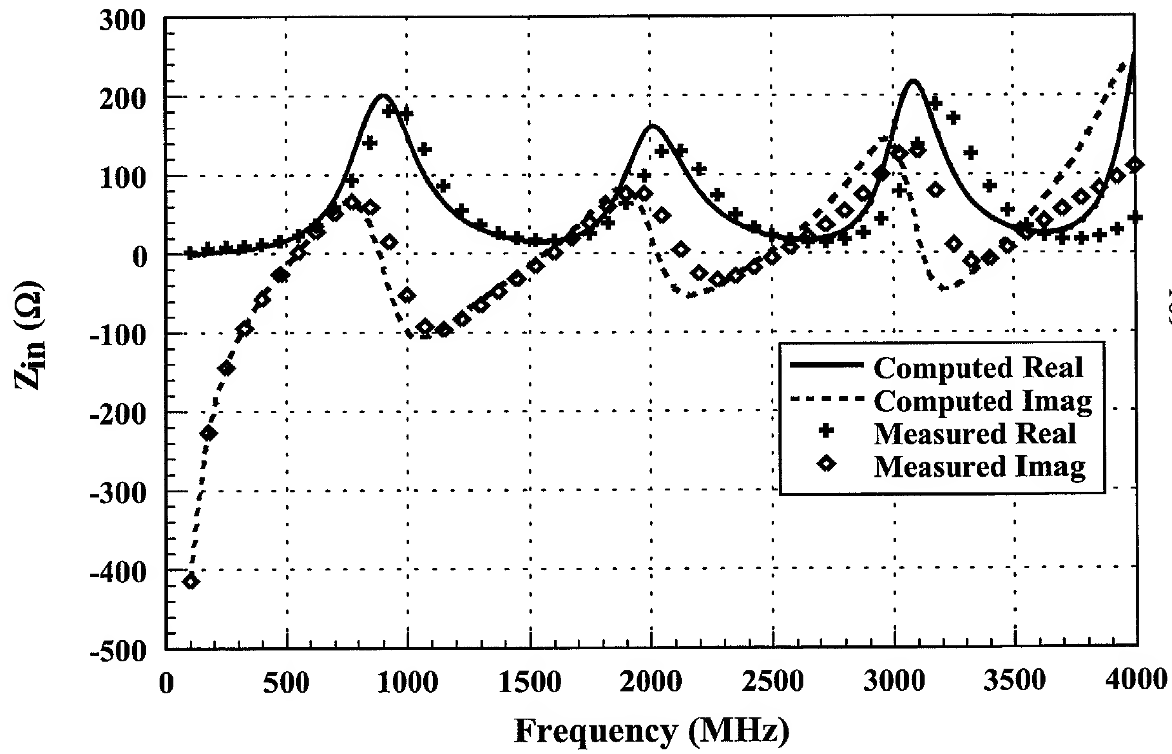


$a = 0.814$ mm, $d = 2$ cm, $w = 3.256$mm, $h_1 = 0.91$ cm, $h_2 = 8.85$ cm

VSWR < 5.0, 475-2750 MHz
BW 5.8:1

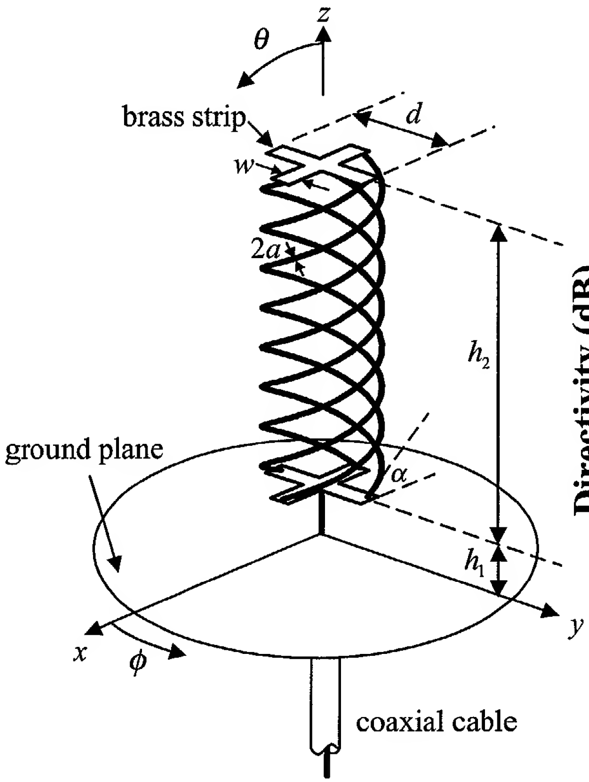
Input Impedance of Quadrifilar Helix

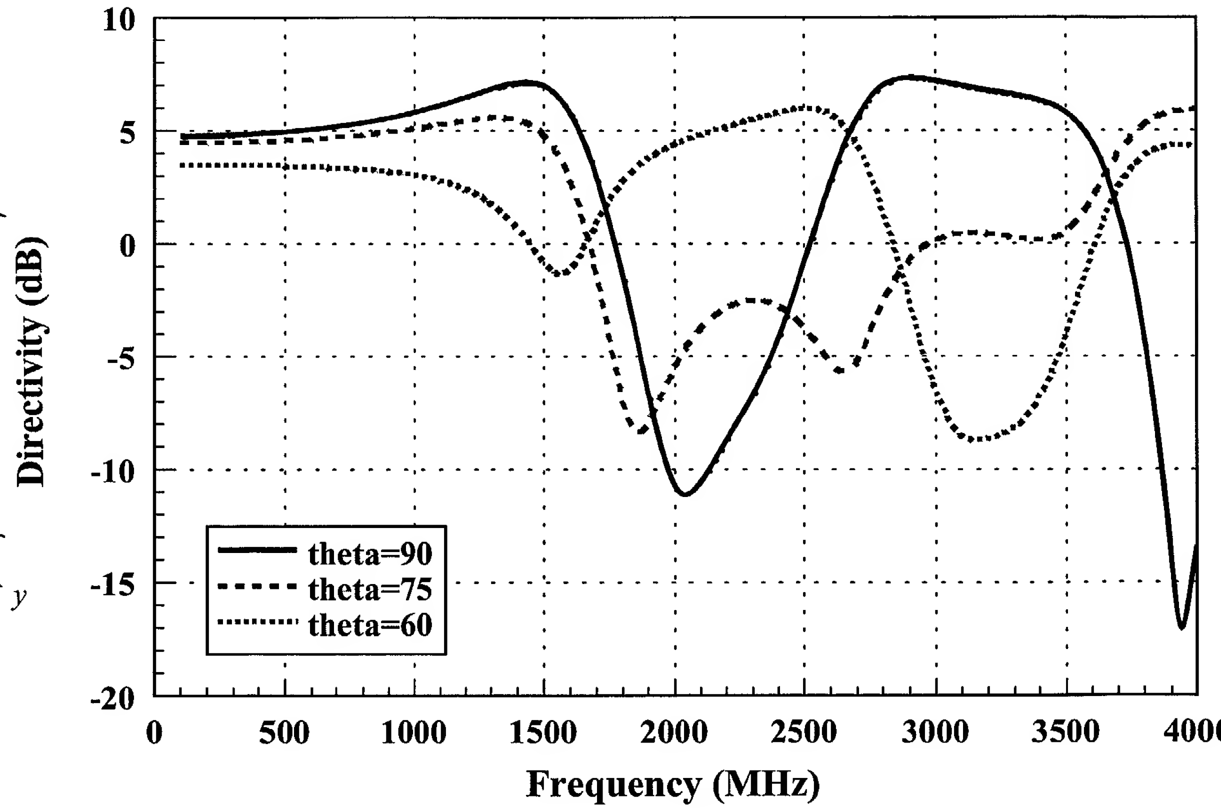




$a = 0.814$ mm, d= 2 cm, w= 3.256mm, h_1 =0.91 cm, h_2 =8.85 cm

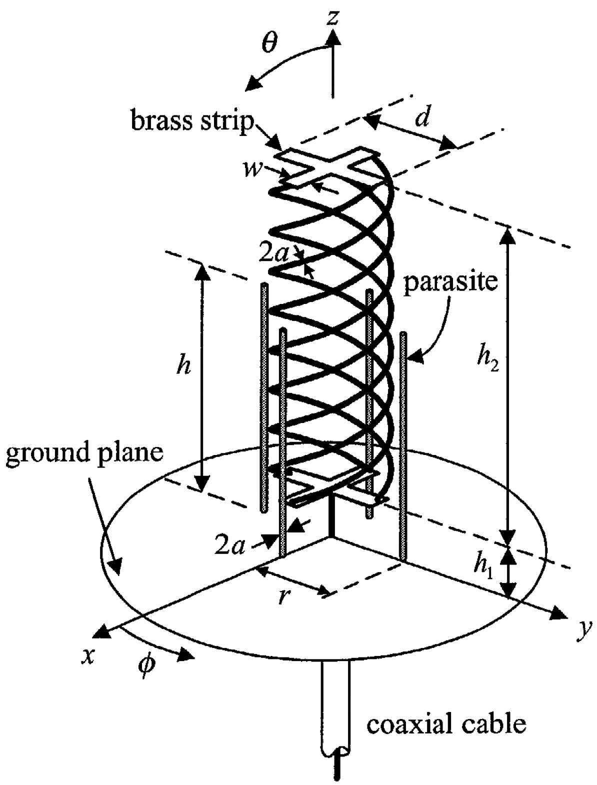
Directivity of Quadrifilar Helix

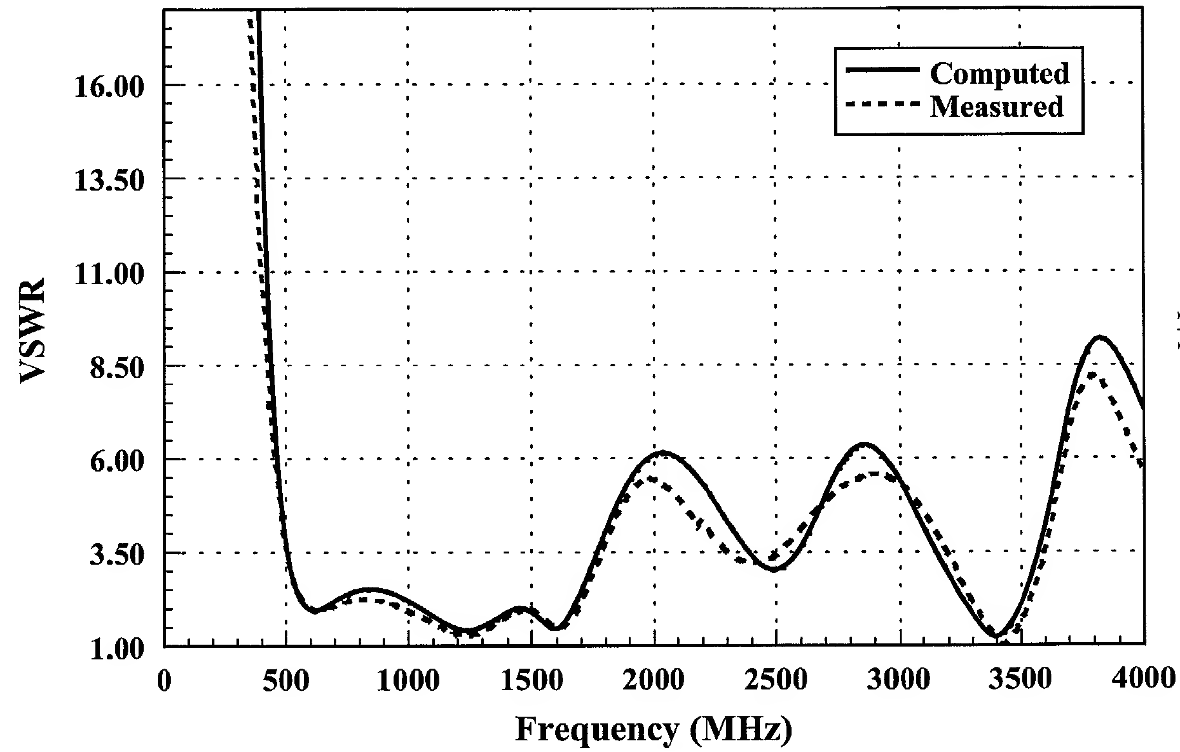




a = 0.814 mm, d= 2 cm, w= 3.256mm, h_1 =0.91 cm, h_2 =8.85 cm

VSWR of Sleeve-Helix

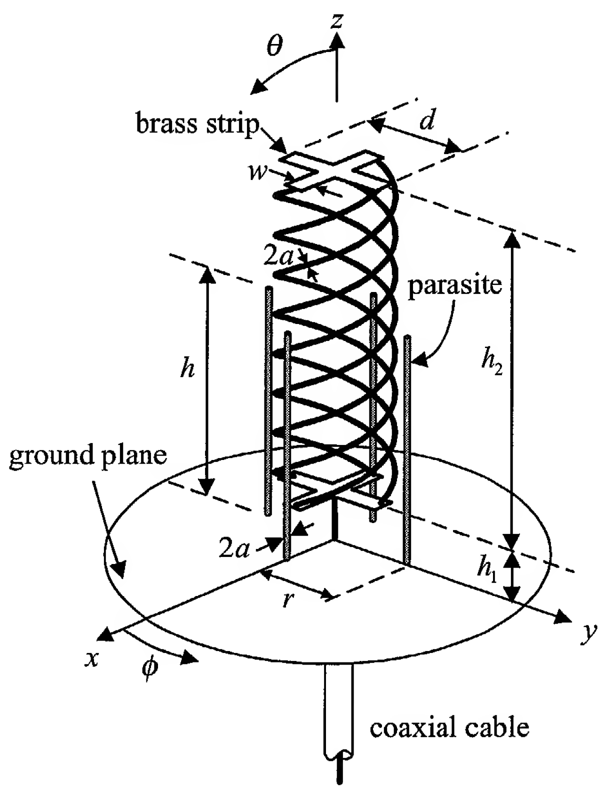


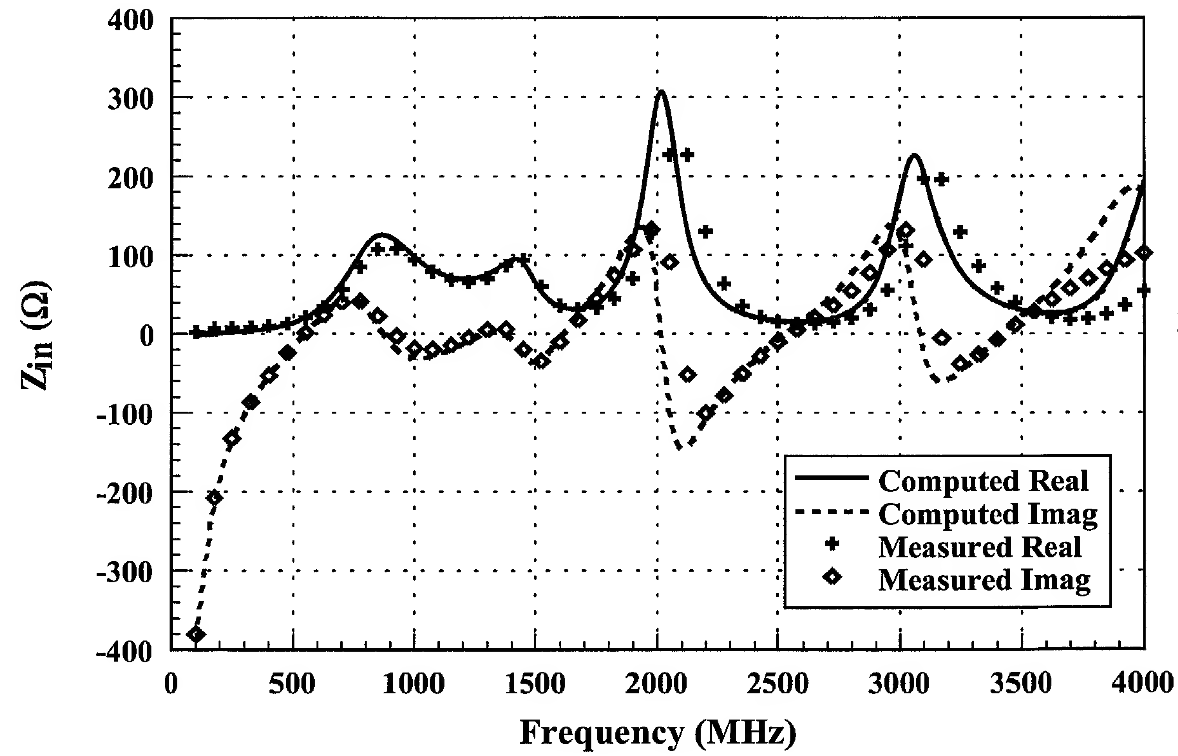


a = 0.814 mm, d= 2 cm, w= 3.256mm, h_1 =0.91 cm, h_2 =8.85 cm, r =3 cm, h=4.76 cm

VSWR < 3.5, 500-1750 MHz
BW 3.5:1

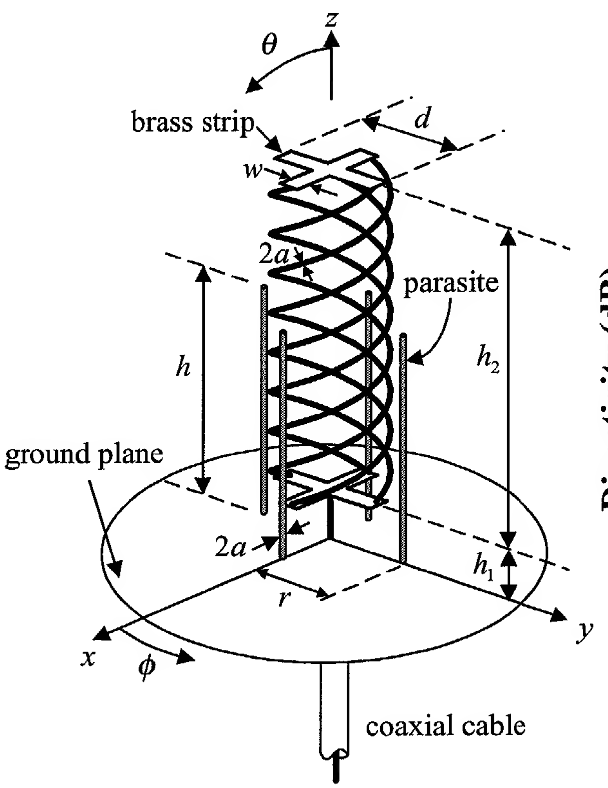
Input Impedance of Sleeve-Helix

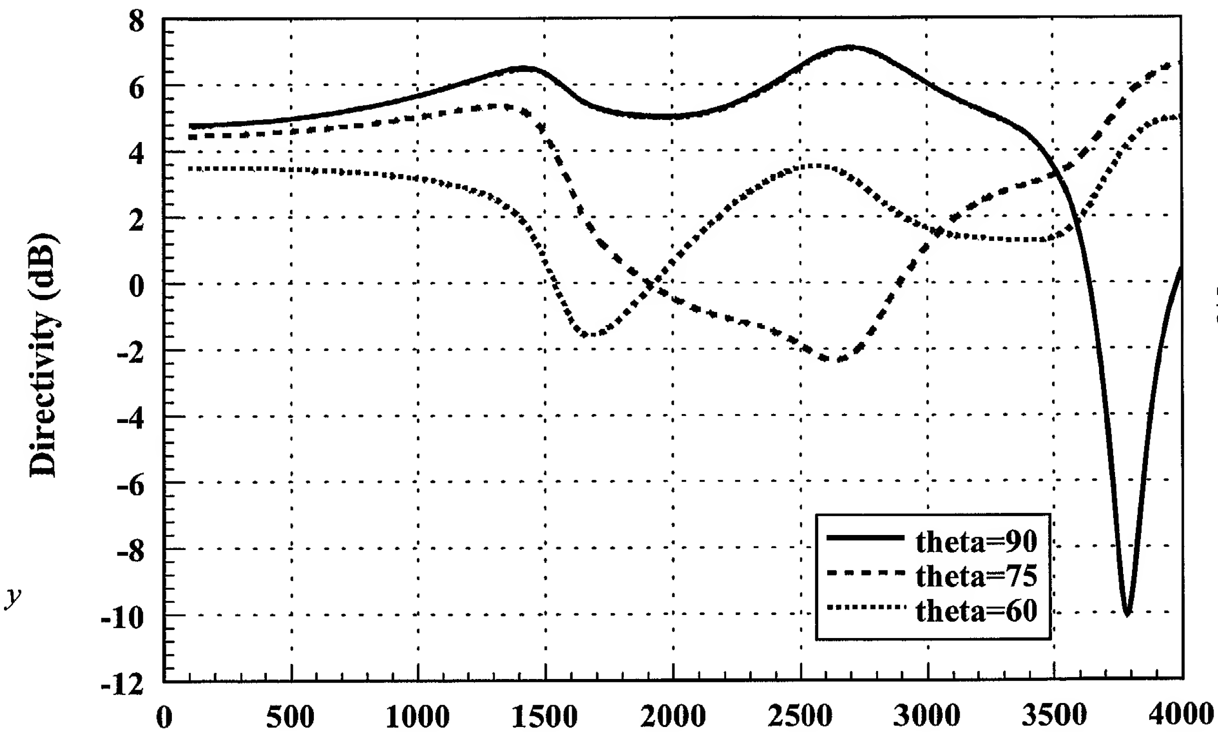




$a = 0.814$ mm, $d = 2$ cm, $w = 3.256$mm, $h_1 = 0.91$ cm, $h_2 = 8.85$ cm, $r = 3$ cm, $h = 4.76$ cm

Directivity of Sleeve-Helix





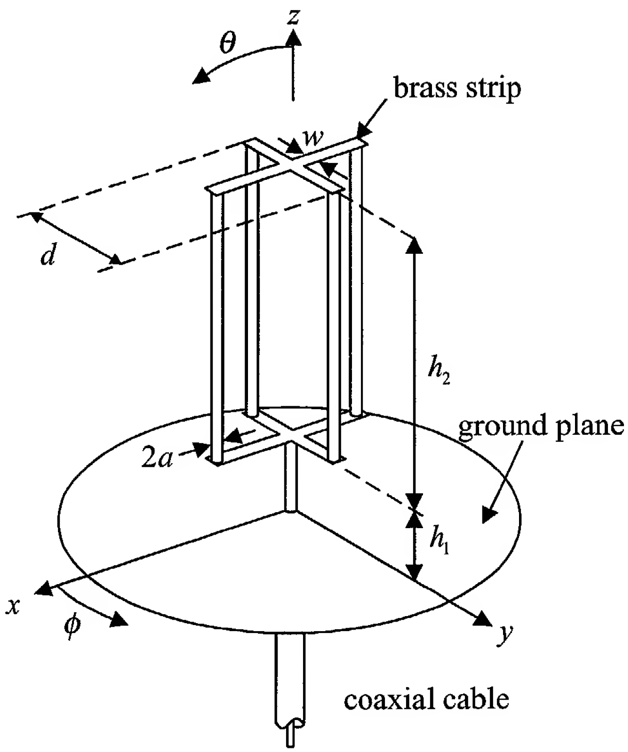
$a = 0.814$ mm, $d = 2$ cm, $w = 3.256$mm, $h_1 = 0.91$ cm, $h_2 = 8.85$ cm, $r = 3$ cm, $h = 4.76$ cm

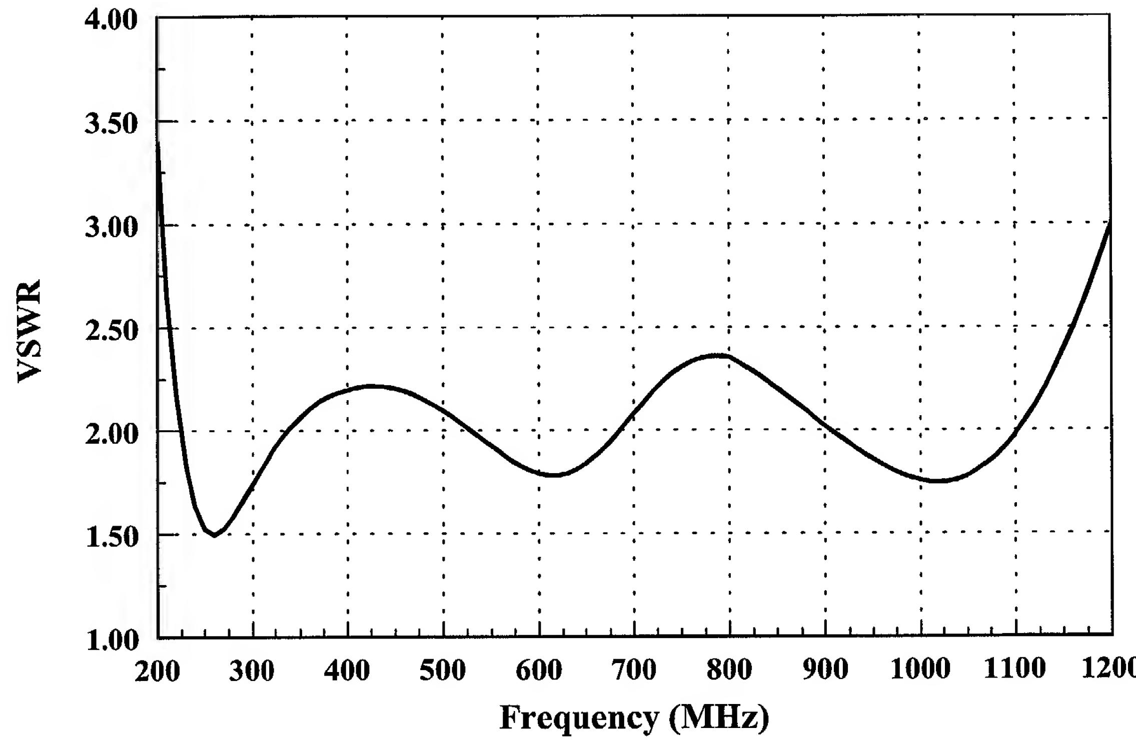
Summary and Comparison of Results VSWR < 3.5

Structure	VSWR	BW Ratio $\frac{f_1}{f_2}$	BW % $100\frac{(f_1-f_2)}{\sqrt{f_1 f_2}}$	Frequency Range (MHZ)	Height (cm)	Width (cm)
Cage monopole	< 3.5	3.0	116	950-2850	17.2	2.2
Sleeve-cage monopole	< 3.5	4.4	163	350-1550	17.2	5
Quadrifilar helix	< 3.5	1.6	47	500-800	9.8	2
Sleeve helix	< 3.5	3.5	134	500-1750	9.8	6
SINCGARS Antenna*	< 3.5	2.9	112	30-88	280	2
Nakano's Helix Monopole	< 3.5	1.7	52	627-1048	19.8 cm	0.4

* Dipole antenna developed and produced by ITT for the Army.

Optimization for VSWR < 2.5

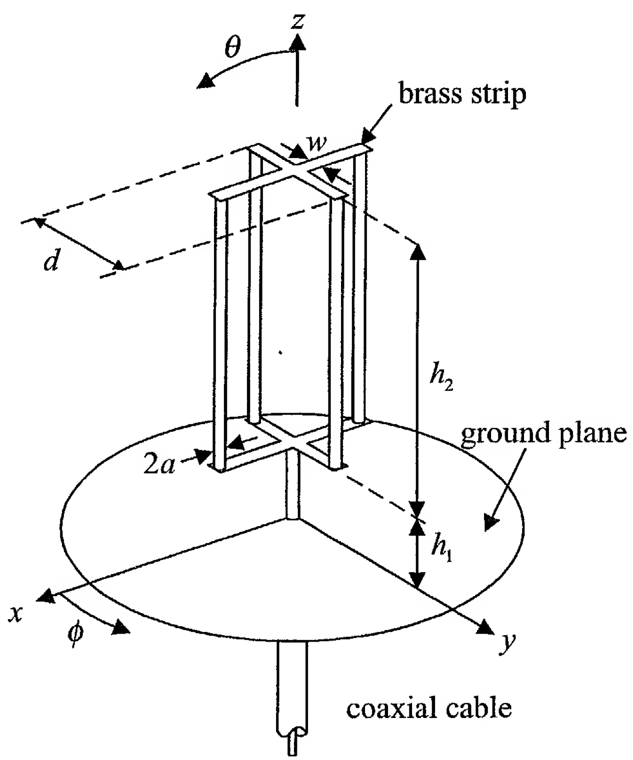


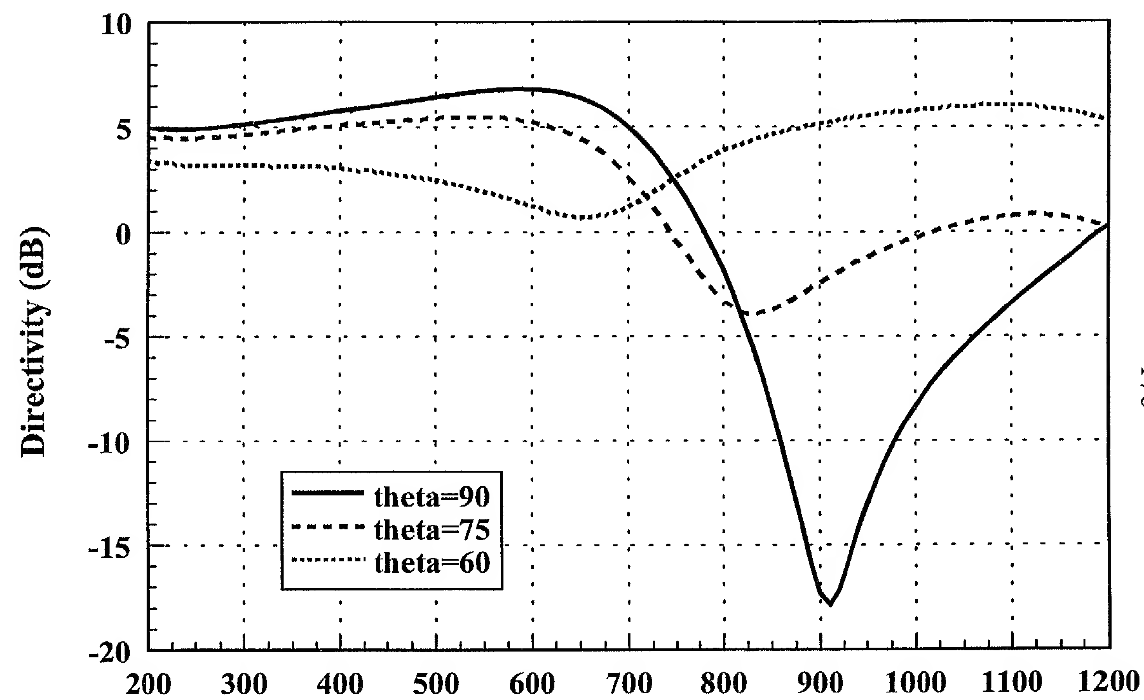


$a = 3.175$ mm, $d = 7.6$ cm, $w = 1.27$ cm, $h_1 = 2.55$ cm, $h_2 = 22.95$ cm.

VSWR < 2.5, 212-1155 MHz
BW 5.5:1

Optimization for VSWR < 2.5





VSWR < 2.5, 212-775 MHz
BW 3.7:1

a = 3.175 mm, d= 7.6 cm, w= 1.27 cm, h_1 =2.55 cm, h_2 =22.95 cm.

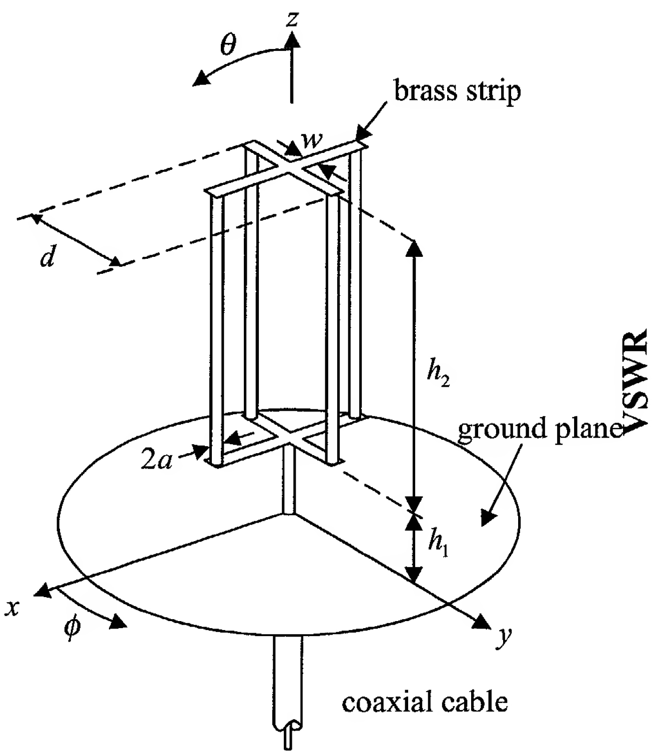
Summary and Comparison of Results for VSWR < 2.5

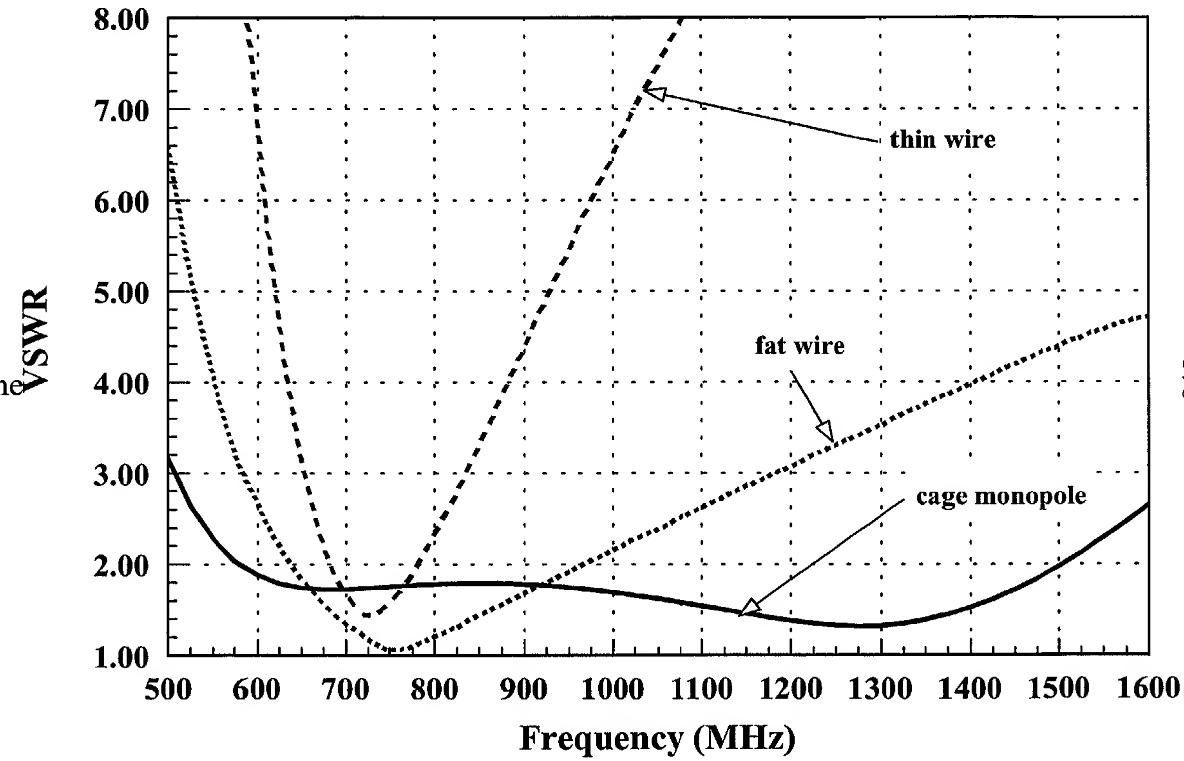
Structure	VSWR	BW Ratio $\frac{f_1}{f_2}$	BW % $100\frac{(f_1-f_2)}{\sqrt{f_1 f_2}}$	Frequency Range (MHz)	Height (cm)	Width (cm)
King's Open Sleeve Dipole	< 2.5	1.77	58.3	225-400	51	13
NTDR Antenna*	< 2.5	2.00	70	225-450	200	6.4
Cage Monopole	< 2.5	3.7	139	212-775	25.5	7.6

Cage was optimized for VSWR < 2.5.

* Dipole antenna developed and produced by ITT for the Army.

Optimization for VSWR < 2.0

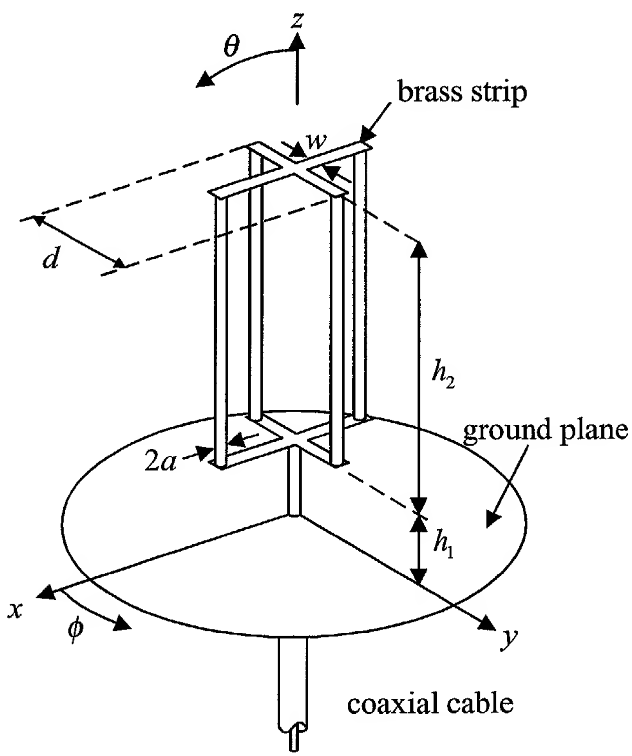


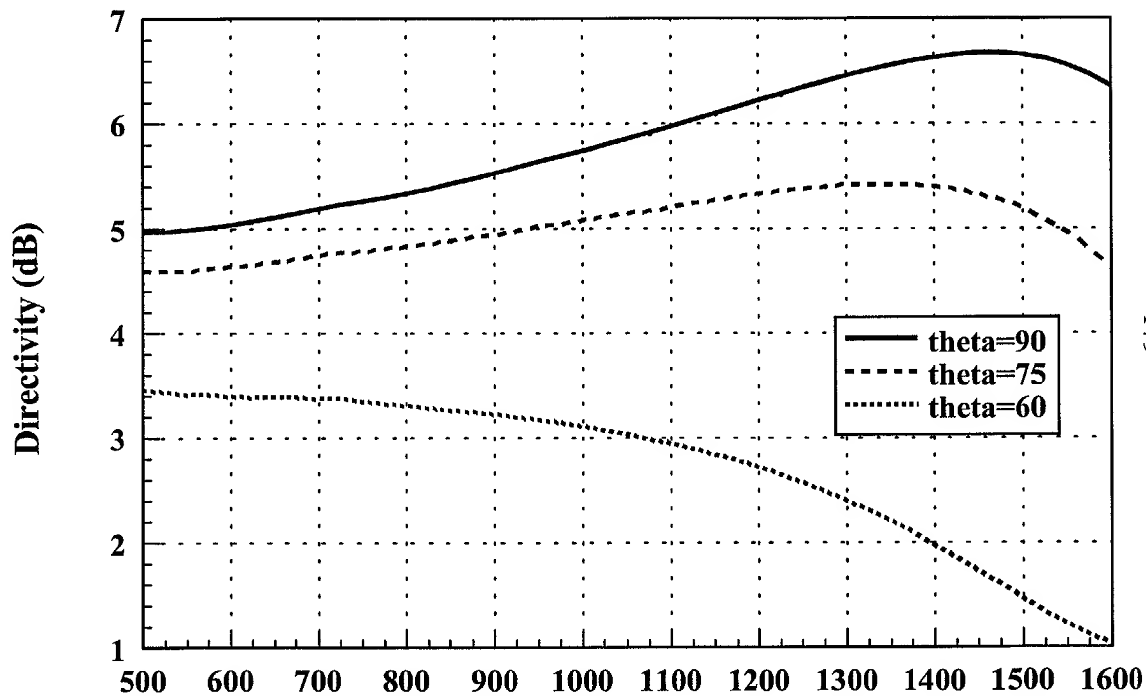


VSWR < 2.0, 575-1500 MHz
BW 2.6:1

a = 0.814 mm, d= 4.8 cm, w= 3.256mm, h_1 =1 cm, h_2 =9 cm.

Optimization for VSWR < 2.0





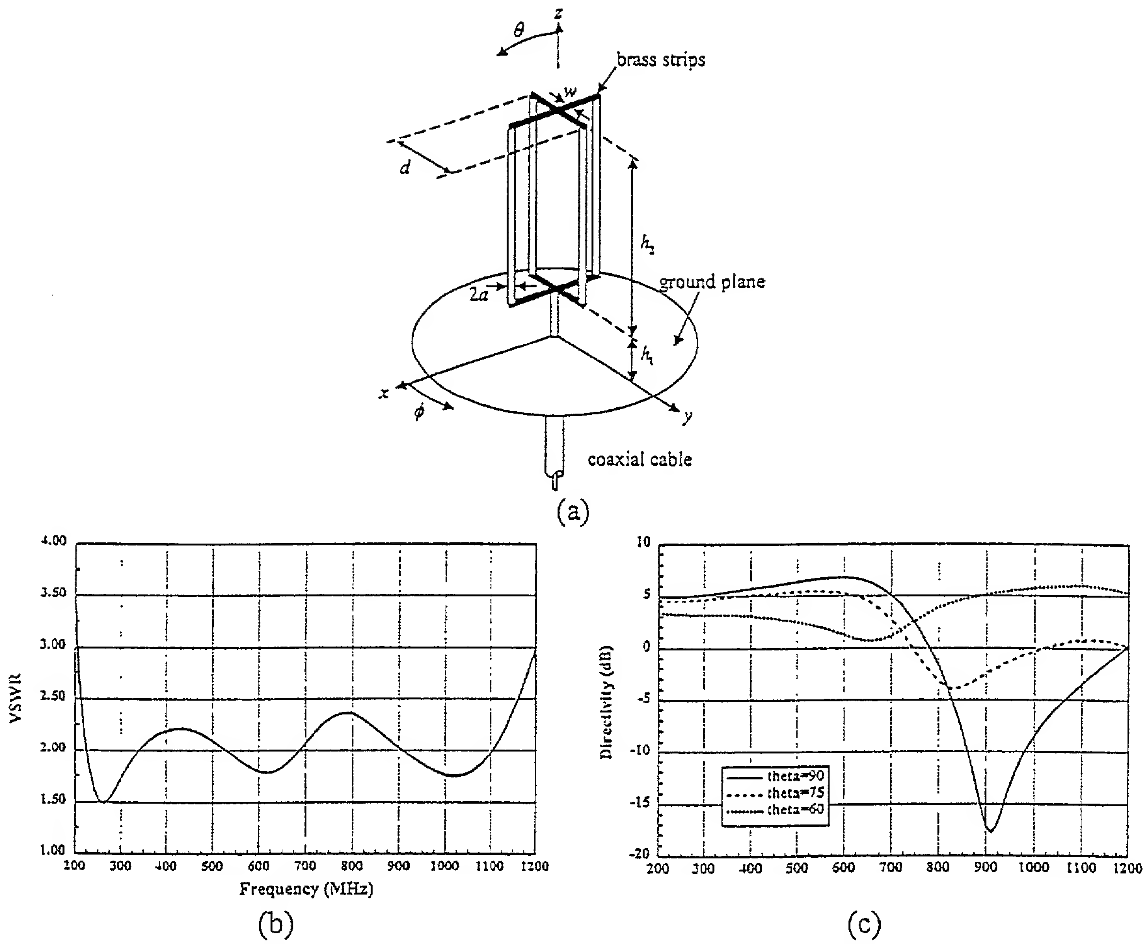
Summary and Comparison of Results for VSWR < 2.0

Structure	VSWR	BW Ratio $\frac{f_1}{f_2}$	BW % $100\frac{(f_1 - f_2)}{\sqrt{f_1 f_2}}$	Frequency Range (MHz)	Height (cm)	Width (cm)	*a* (mm)
Nakano's Helix-Monopole	< 2.0	1.14	13.44	662-757	19.8	0.4	0.015, 0.003
	< 2.0	1.05	5.78	957-1014			
Cage Monopole	< 2.0	2.60	99.6	575-1500	10	4.8	0.814

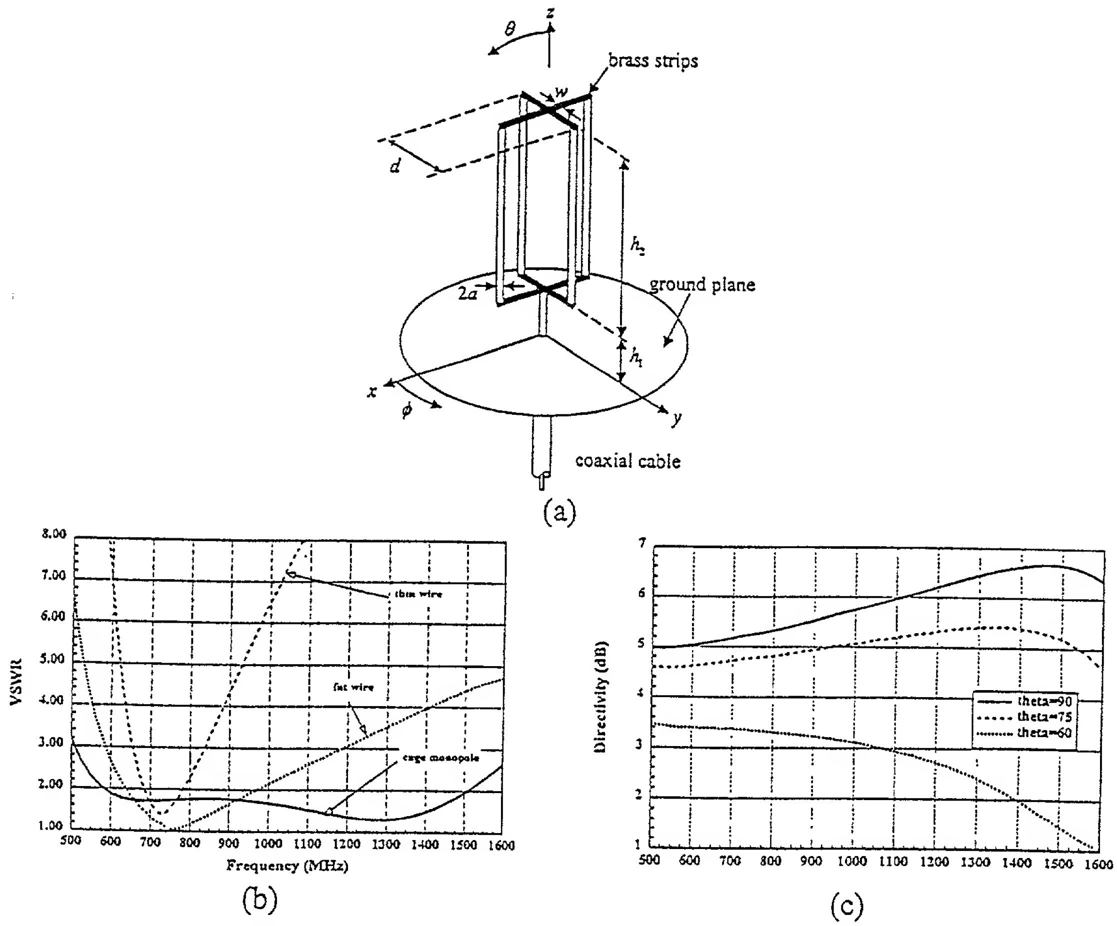
Cage was optimized for VSWR < 2.0.

Conclusions

- Cage structures can be optimized for lower VSWR.
- Parasites of optimum size and placement improve VSWR of driven antenna.
- Helical elements reduce height at sacrifice of bandwidth
- Wire radius is an important parameter.



Data for cage monopole optimized for VSWR < 2.5 ($a = 3.175$mm, $d = 7.6$cm, $w = 1.27$cm, $h_1 = 2.55$ cm, $h_2 = 22.95$ cm) (a) VSWR, (b) directivity.



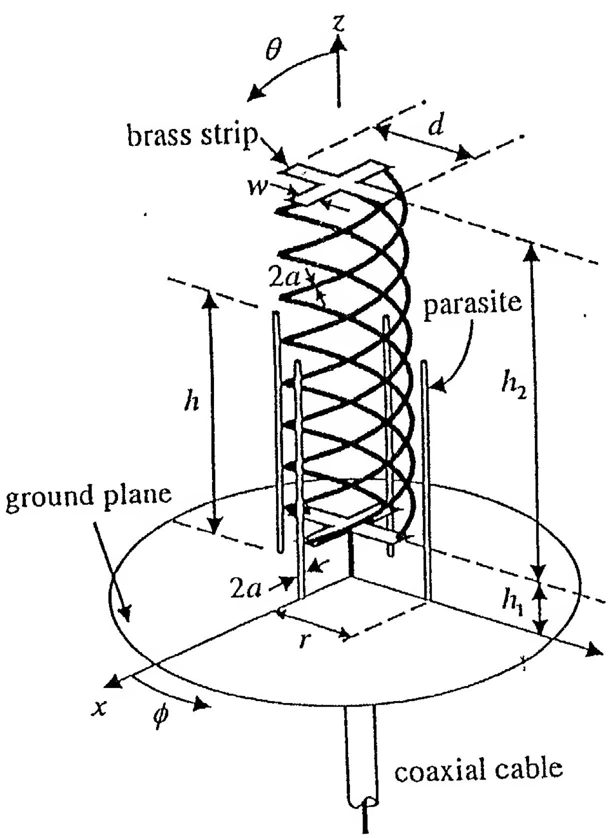
Data for cage antenna ($a = 0.814$mm, $d = 4.8$cm, $w = 3.256$mm, $h_1 = 1$cm, $h_2 = 9$cm) optimized for VSWR < 2, thin wire (a=0.814mm, h=10 cm), and fat wire (a=4.8 cm, h=10 cm). (a) VSWR, (b) directivity of cage.

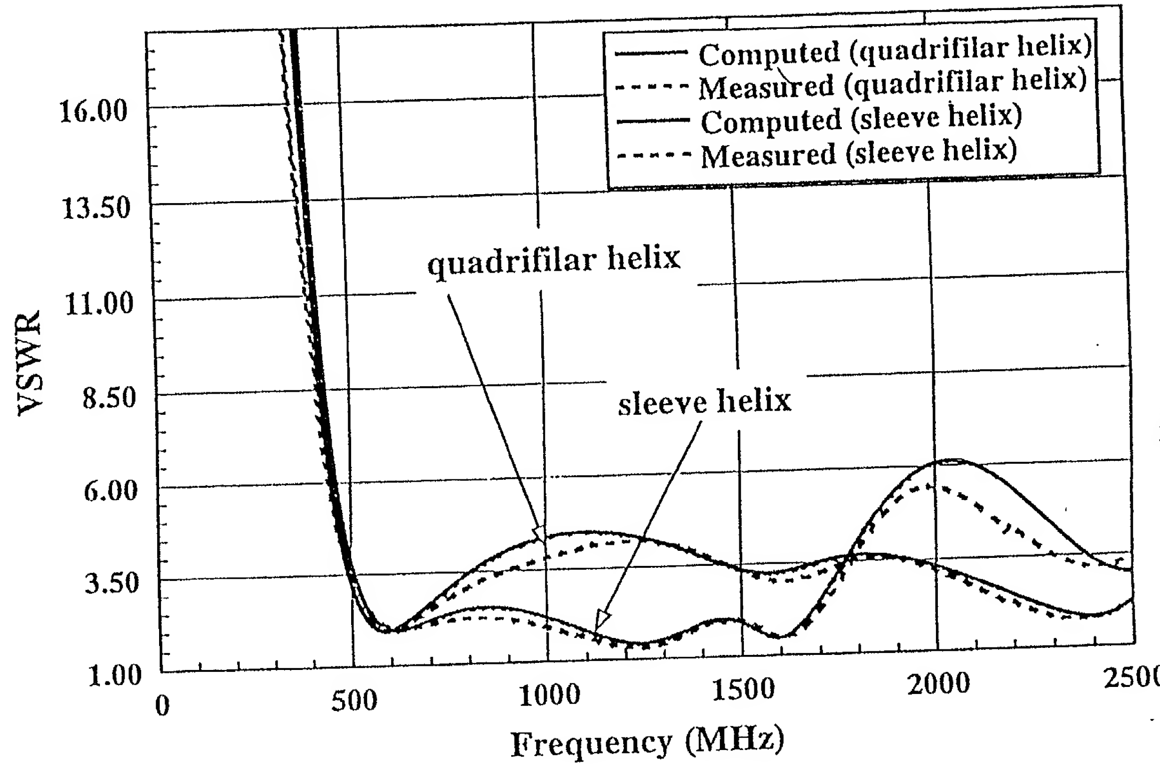
Summary and Comparison

Structure	VSWR	BW Ratio $\frac{f_1}{f_2}$	BW % $100\frac{(f_1-f_2)}{\sqrt{f_1 f_2}}$	Frequency Range (MHZ)	Height (cm)	Width (cm)
Cage monopole	< 3.5	3.0	116	950-2850	17.2	2.2
Sleeve-cage monopole	< 3.5	4.4	163	350-1550	17.2	5
Quadrifilar helix	< 3.5	1.6	47	500-800	9.8	2
Sleeve helix	< 3.5	3.5	134	500-1750	9.8	6
SINCGARS Antenna*	< 3.5	2.9	112	30-88	280	2
Nakano's Helix Monopole	< 3.5	1.7	52	627-1048	19.8 cm	0.4

* Dipole antenna developed and produced by ITT for the Army.

VSWR of Sleeve-Helix





$a = 0.814$ mm, $d = 2$ cm, $w = 3.256$mm, $h_1 = 0.91$ cm, $h_2 = 8.85$ cm, $r = 3$ cm, $h = 4.76$ cm

VSWR < 3.5, 500-1750 MHz
BW 3.5:1